



UNIVERSIDAD
POLITECNICA
DE VALENCIA

TESIS DOCTORAL

**POSTPROCESAMIENTO CAM-ROBOTICA ORIENTADO
AL PROTOTIPADO Y MECANIZADO EN CELULAS
ROBOTIZADAS COMPLEJAS**

*(CAM-ROB Postprocessing aimed at Prototyping and Machining in
complex robotic workcells)*

Presentada por: Javier Andrés de la Esperanza

*Dirigida por: Josep Tornero i Montserrat
Luis Gracia Calandín*

Valencia, Marzo de 2011

Quiero agradecer al director de la tesis Dr. D. Josep Tornero i Montserrat la oportunidad de desarrollar la tesis en las instalaciones del Instituto de Diseño y Fabricación (IDF). Al también director de la tesis Dr. D. Luis Gracia Calandín quiero reconocer su valioso, continuado y efectivo asesoramiento, su estímulo y su paciencia.

También quiero agradecer su motivación y consejos al Dr. D. Juan Antonio García Manrique, ya desde los tiempos en que empecé a cursar Ingeniería en la UPV. Además, los profesores Dr. D. Carlos Gracia Calandín y Dr. D. Francisco González Contreras me han aconsejado certeramente en los coletazos finales de esta tesis. A D. Guillermo Bruixola Casani por su desinteresada asistencia en el manejo del NXTM, cada vez que lo he requerido.

A Marta Gallart Penalva por aguantar y animarme en los momentos arduos. A mis amigos Miguel García Ponce, Alejandro Cerdá Dols, Antonio Astorgano Lozano, Isaac Suárez Alvarez y Gilberto González Parra por los momentos de evasión deportiva.

Finalmente, vaya mi más sincero agradecimiento a mis compañeros y excompañeros en el IDF.

A mi familia

RESUMEN

El principal interés de la presente tesis consiste en el estudio e implementación de *postprocesadores* para adaptar las trayectorias generadas por sistemas de Fabricación Asistida por Computador (generalmente conocidos como plataformas CAM, *Computer Aided Manufacturing*) hacia una célula robotizada de ocho articulaciones, la cual está destinada al prototipado de piezas 3D diseñadas desde plataformas CAD (*Computer Aided Design*). Dicha célula la conforma un robot manipulador industrial de seis articulaciones rotativas, el cual está montado sobre un rail y sincronizado con una mesa giratoria. Para alcanzar el objetivo principal expuesto inicialmente, sucesivas tareas son llevadas a cabo. Cada una de éstas conlleva una metodología, objetivo y resultados parciales que se conjugan y complementan, a saber:

- Se describe la arquitectura de la célula a niveles de posición y velocidad articulares para las resoluciones directa e inversa en ambos casos. El condicionamiento numérico de la matriz Jacobiana se describe como *índice kinetostático* para evaluar la cercanía a configuraciones singulares. Éstas son analizadas desde un punto de vista geométrico.
- Previo a cualquier mecanizado, las articulaciones externas adicionales requieren de una calibración realizada *in situ*, generalmente en el lugar de trabajo. Se ha desarrollado un novedoso método de *Calibración sin contacto en base a restricciones planares* para estimar los parámetros de configuración de las articulaciones externas, por medio de un sensor láser de desplazamiento.
- Un primer control, a nivel de desplazamiento por medio de un motor de inferencia borrosa, es integrado en el postprocesador del sistema CAM.
- Varios Esquemas de Resolución de Redundancias a nivel de velocidad articular son comparados para la configuración de un postprocesador. Estos esquemas tratan no solo con las articulaciones adicionales (*redundancia intrínseca*) sino también con la redundancia debida a la simetría de la herramienta de corte (*redundancia funcional*).
- El uso de estos esquemas es optimizado mediante el ajuste de dos vectores de criterio de comportamiento (*performance criterion vectors*) relacionados con la *evitación de singularidades* y el mantenimiento de una *postura de referencia* preferente. Dos novedosos motores de inferencia borrosa ajustan activamente el peso (o relevancia) de cada articulación en estas tareas.

El sistema completo resultante es validado en el prototipado real de un *modelo orográfico* y de una *Falla Valenciana*.

ABSTRACT

The main interest of this thesis consists of the study and implementation of *postprocessors* to adapt the toolpath generated by a *Computer Aided Manufacturing* (CAM) system to a complex robotic workcell of eight joints, devoted to the rapid prototyping of 3D CAD-defined products. It consists of a 6R industrial manipulator mounted on a linear track and synchronized with a rotary table. To accomplish this main objective, previous work is required. Each task carried out entails a methodology, objective and partial results that complement each other, namely:

- It is described the architecture of the workcell in depth, at both *displacement* and *joint-rate* levels, for both *direct* and *inverse* resolutions. The conditioning of the Jacobian matrix is described as *kinetostatic performance index* to evaluate the vicinity to *singular postures*. These ones are analysed from a geometric point of view.
- Prior to any machining, the additional external joints require a calibration done *in situ*, usually in an industrial environment. A novel *Non-contact Planar Constraint Calibration* method is developed to estimate the external joints configuration parameters by means of a laser displacement sensor.
- A first control is originally done by means of a fuzzy inference engine at the displacement level, which is integrated within the postprocessor of the CAM software.
- Several Redundancy Resolution Schemes (RRS) at the joint-rate level are compared for the configuration of the postprocessor, dealing not only with the additional joints (*intrinsic redundancy*) but also with the redundancy due to the symmetry on the milling tool (*functional redundancy*).
- The use of these schemes is optimized by adjusting two performance criterion vectors related to both *singularity avoidance* and maintenance of a preferred *reference posture*, as secondary tasks to be done during the path tracking. Two innovative fuzzy inference engines actively adjust the weight of each joint in these tasks.

The resulting whole system is validated in a real prototyping of an *orographic model* and a *Valencian Falla*.

El principal interès de la present tesi consisteix en l'estudi i implementació de *postprocesadors* per a adaptar les trajectòries generades per sistemes de Fabricació Assistida per Computador (normalment conegudes com a plataformes CAM, *Computer Aided Manufacturing*) cap a una cèl·lula robotitzada de huit articulacions, la qual està destinada al prototipat ràpid de peces 3D dissenyades des de plataformes CAD (*Computer Aided Design*). Aquesta cèl·lula la conforma un robot manipulador industrial de sis articulacions rotatives, el qual està muntat sobre un rail i sincronitzat amb una taula giratòria. Per a aconseguir l'objectiu principal exposat inicialment, successives tasques són dutes a terme. Cadascuna d'estes comporta una metodologia, objectiu i resultats parcials que es conjuguen i complementen, a saber:

- Es descriu en profunditat l'arquitectura de la cèl·lula, a nivells de posició i velocitat articulars, per a les resolucions directa i inversa en ambdós casos. El condicionament numèric de la matriu Jacobiana es descriu com índex kinetostatic per a avaluar la proximitat a configuracions singulars. Estes són analitzades des d'un punt de vista geomètric.
- Previ a qualsevol mecanitzat, les articulacions externes addicionals requereixen d'una calibració realitzada in situ, generalment en el lloc de treball. S'ha desenrotllat un nou mètode de *Calibració sense contacte amb restriccions planars* per a estimar els paràmetres de configuració de les articulacions externes, per mitjà d'un sensor làser de desplaçament.
- Un primer control, desenrotllat originàriament a nivell de desplaçament per mitjà d'un motor d'inferència borrosa (*fuzzy*), és integrat en el postprocesador del sistema CAM.
- Diversos Esquemes de Resolució de Redundàncies a nivell de velocitat articular són comparats per a la configuració d'un postprocesador. Aquests esquemes tracten no sols amb les articulacions addicionals (*redundància intrínseca*) sinó també amb la redundància deguda a la simetria de la ferramenta de tall (*redundància funcional*).
- L'ús d'aquests esquemes és optimitzat per mitjà de l'ajust de dos vectors de criteri de comportament (*performance criterion vectors*) relacionats amb l'evitació de singularitats i el manteniment d'una postura de referència preferent. Dos nous motors d'inferència borrosa ajusten activament el pes (o relevància) de cada articulació en aquestes tasques.

El sistema complet resultant és validat en el prototipat real d'un model orogràfic i d'una Falla Valenciana.

CONTENTS

CHAPTER 1. INTRODUCTION	29
1.1. INTRODUCTION.....	29
1.2 STATE OF ART AND CURRENT TENDENCIES	31
1.3 OBJETIVES	32
1.4 METHODOLOGY	33
1.5 STRUCTURE.....	34
References (Ch. 1).....	36
CHAPTER 2. WORKCELL KINEMATIC CHARACTERIZATION.....	39
2.1. CONCEPTS ON MANIPULATOR KINEMATICS	39
2.1.1. Joint variables (generalized coordinates)	40
2.1.2. Operational coordinates.....	41
2.2. LEVEL OF KINEMATIC ANALYSIS	42
2.2.1. Direct and Inverse Kinematic Problem at the displacement level. .	43
2.2.2. Kinematic analysis at joint-rate level.	46
i) DKP at joint-rate level.	46
ii) IKP at joint-rate level.....	46
iii) The Jacobian matrix.....	48
2.2.3. Singular configurations.	51
i) Consideration to wrist-partitioned manipulator singularities	52
2.3. KINETOSTATIC PERFORMANCE INDICES. POSTURE- DEPENDENT INDICES.....	56
2.3.1. Manipulability	57
2.3.2. Condition number of J	60
i) Inhomogeneity of J , and Characteristic length	60
ii) Formula for the condition number of J	61
iii) Consideration to wrist-partitioned manipulator singularities	62
2.4. KINEMATIC CHARACTERIZATION OF AN INDUSTRIAL WORKCELL.....	63
2.4.1. Components of the numerically controlled KUKA workcell.....	64

i) KUKA KRC2 controller.....	64
ii) KUKA KR15/2 manipulator	65
iii) Additional linear axis	66
iv) Additional rotary table.....	66
2.4.2. Direct Kinematic model of position.....	67
2.4.3. Inverse Kinematic Problem (IKP) of position	71
i) Geometric approach for the IKP of position	72
ii) Resolution of the gross positioning	73
iii) Resolution of the fine positioning	74
2.4.4. Workcell Jacobian.....	78
2.4.5. Tool-holder characterization.....	80
2.4.6. Characteristic length L of the KUKA KR-15/2	83
References (Ch. 2)	86
CHAPTER 3. WORKCELL CALIBRATION	91
3.1. CONCEPTS ON ACCURACY CRITERIA AND ERROR SOURCES	91
.....	
91	
3.2. CALIBRATION. MATHEMATICAL BACKGROUND.....	96
3.2.1. Problem statement.....	96
3.2.2. Solving the least-squares problem	97
i) Linear least-squares	98
ii) Non-Linear Least-Squares (NLSQ).....	99
3.2.3. Gauss-Newton algorithm and its application to calibration	100
algorithms	100
i) Description of the iterative method	101
ii) Outline of the NLSQ Model-based calibration methods. 2D planar	103
calibration example.....	103
3.3. CALIBRATION OF THE ADDITIONAL JOINTS OF THE KUKA	111
WORKCELL	111
3.3.1. Non-contact Planar Constraint Calibration procedure. Material and	111
method	111
i) Laser displacement sensor.....	113
3.3.2. Formulations	115
i) Formulation of the Kinematic Identification Model	115
3.3.3. Results.....	119
References (Ch. 3)	121
CHAPTER 4. CAM TO WORKCELL POSTPROCESSING	127

4.1. INTEGRATED PRODUCTION SYSTEMS.....	127
4.1.1. Benefits of the integrated production systems	128
4.2. COMPUTER NUMERICAL CONTROL (CNC).....	130
4.2.1. Definition	131
4.2.2. Classification of the Numerical Control systems	131
i) Interpolation.....	133
4.3. CAM SYSTEMS FOR TOOLPATH generaTiOn.....	134
4.4. POSTPROCESSING.....	134
4.4.1. Concept of postprocessing	135
4.4.2. Literature review in CNC Postprocessing	137
4.4.3. CAM-ROB postprocessing	143
4.5. NX-CAM TOOLPATH GENERATION.....	144
4.5.1. NX-CAM module characteristics.....	144
i) Trajectory generation (CL-File). Linear and circular path tracking	144
ii) NX TM -Post	147
4.6. INDUSTRIAL NX TM TO KUKA TM WORKCELL POSTPROCESSING.....	149
4.6.1. KUKA TM Workcell programming.....	149
4.6.2. KRL for PTP motions (synchronous PTP).....	150
4.6.3. KUKA KRL for Continuous Path Tracking.....	151
4.6.4. Post programming	153
References (Ch. 4).....	154
CHAPTER 5. Redundancy resolution schemes	159
5.1. KINEMATIC REDUNDANCY	159
5.1.1. Definition of Kinematic Redundancy	159
i) Intrinsic redundancy	161
ii) Functional redundancy.....	161
5.2. CONTINUOUS PATH PLANNING AND TRACKING.....	163
5.3. REDUNDANCY RESOLUTION SCHEMES (RRS).....	167
5.3.1. Local Optimization Algorithms for intrinsically-redundant manipulators (r_I)	169
i) Schemes with the Moore-Penrose Pseudo-Inverse.....	169
ii) Schemes using the Weighted Pseudo-Inverse	172
iii) Schemes using Householder Reflection	173
iv) Schemes using the damped least-squares (DLS-) inverse	175
5.3.2. Solution of functionally-redundant manipulators (r_F).....	177
i) Virtual Joint Method (VJM)	178
ii) Twist Decomposition Method (TDM).....	179

5.3.3. Consideration for functionally-redundant (r_F) and intrinsically-redundant (r_I) manipulators.....	180
5.3.4. Redundant manipulator controlling process	183
5.4. INTELLIGENT CONTROL IN REDUNDANCY RESOLUTION	183
5.4.1. Fuzzy-Based Redundancy-Resolution Approaches.....	185
5.4.2. Fuzzy Logic Overview.....	186
i) Fuzzy sets and membership functions.....	187
ii) Logical Operations	188
iii) If-Then Rules	189
iv) Fuzzy Inference Process.....	189
References (Ch. 5)	192
CHAPTER 6. ANALYSIS AND RESULTS	199
6.1. INTRODUCTION	199
6.2. FUZZY LOGIC FOR IK POSITIONING (IKP) PROBLEM	200
6.2.1. Development of the fuzzy controller	202
i) Variable definition.....	202
ii) Clusterization of input and output spaces.....	203
iii) Fuzzification of the input variables.....	203
iv) Knowledge base.....	203
v) Inference engine	204
vi) Defuzzification	205
vii) Interactivity with fuzzy controller module.....	206
6.2.2. Analysis and results	207
6.3. IK PROBLEM IMPLEMENTATION AT RATE LEVEL	208
6.3.1. Discussion.....	208
6.3.2. TDM and VJM test implementation	209
i) Method.....	209
ii) Performance criterion vector, h	212
iii) Algorithm for the VJM.....	215
iv) Algorithm for the TDM.....	216
6.3.3. Analysis and results.	218
i) Constant weighting vector for the combined performance criterion... ..	218
ii) Adapted Fuzzy weighting vector for the combined performance criterion.....	226
6.3.4. Periodic revision by fuzzified IK analysis.....	234
6.3.5. Comparison of the previous VJM improvements	237
References (Ch. 6)	239

CHAPTER 7. APPLICATIONS	243
7.1. INTRODUCTION.....	243
7.2. MATERIAL AND METHODS	243
7.2.1. Material	243
i) Software	243
ii) Hardware.....	245
7.2.2. Methods.....	246
7.3. CASES STUDIED	248
7.3.1. Orographic model.....	249
7.3.2. Valencian <i>Ninot</i>	243
References (Ch. 7).....	262
CHAPTER 8. CONCLUSIONS AND FUTURE WORK	265
8.1. CONCLUSIONS.....	265
8.2. FUTURE WORK	267

LIST OF TABLES

Table 2.1. <i>Range of motion</i> referred to the mechanical zero of the robot axis concerned. It is limited by means of software switches on all axes.	65
Figure 2.20. and Table 2.2. Frame assignments and parameters for the KUKA KR15/2 with the standard DH. The posture shown corresponds to a commonly used HOME position ($\theta_i = \{0, -\pi/2, 0, 0, \pi/2, 0\}$ in this model, for $i=1, \dots, 6$).....	69
Figure 2.21. and Table 2.3. Frame assignments and parameters for the complete milling workcell at the IDF, formed by the KR15/2 manipulator mounted on the linear track and synchronized with the rotary table. The posture shown corresponds to a commonly used HOME position (in this model, $\theta_i = \{ \pi, \pi, -\pi/2, 0, 0, \pi/2, 0\}$ for $i=M, 1, 2, \dots, 6$; and $d_L=0$)	70
Table 2.4. Position and orientation measured for the tool-holder regarding the robot flange frame.	81
Figure 2.28. and Table 2.5. Robot {EE} frame assignments and parameters for the complete DH model of the IDF workcell, with the tool holder designed for milling purposes.	82
Table 3.1. DH parameters for the mechanical system simulating the manipulator gross positioning.	104
Table 3.2. MDH Matrix simulating the manipulator gross positioning.....	110
Table 3.3. MDH Matrix simulating the manipulator gross positioning.....	111
Table 4.1. Offset between the mechanical and the DH modelled joint values .	151
Table 4.2. NX Event Handler variables for circular movements.....	153
Table 5.1. Table summarizing the parameters for both standard DH-models.	182

Table 6.1. Knowledge base for $\Delta\theta_M$	204
Table 6.2. Knowledge base for Δd_L	204
Table 6.3. Case studied	207
Table 6.4. From left (element $(1,1)$) to right (element $(8,8)$), diagonal weighting matrixes for the combined performance criterion.	218
Table 6.5. Experimental results for the simulation in the studied workcell of the VJM and TDM algorithms. In the TDM, two variations including a projection in $\aleph(J)$ are studied, according to case (c).	221
Table 6.6. From left (element $(1,1)$) to right (element $(8,8)$), diagonal weighting matrixes for the combined performance criterion.	222
Table 6.7. Experimental results for the simulation in the studied workcell of the VJM and TDM algorithms, with the constant weights of Table 6.6. In the TDM, two variations including a projection in $\aleph(J)$ are studied, according to case (c).	225
Table 6.8. From left (element $(1,1)$) to right (element $(8,8)$), diagonal weighting matrixes for the combined performance criterion. The fuzzyfied weights are assigned to the most significant joints according to experience.	227
Table 6.9. Experimental results for the simulation in the studied workcell of the VJM and TDM algorithms, with the adapted weights of Table 6.8 via fuzzy inference. In the TDM, two variations including a projection in $\aleph(J)$ are studied, according to case (c).	233
Table 6.10. Experimental results for the simulation in the studied workcell of the VJM with the implementation of the adapted fuzzy weighting vector and a periodic IKP position analysis as depicted in Figure 6.16.	236
Table 7.1. Range of motion and conditioning of the manipulator while the execution of the task.	252
Table 7.2. Range of motion and conditioning of the manipulator while the execution of the task.	259

LIST OF FIGURES

Figure 2.1 Left, Prismatic (P) and Revolute (R) joints at an industrial workcell. Right, detail of the construction of a revolute joint	39
Figure 2.2. Generalized coordinates in a planar manipulator	40
Figure 2.3. Left, coordinate system {E} referred to {B} by means of ${}^B T_E$. Right, conventionalism in the specific case of the terminal organ of the robot.	42
Figure 2.4. Mapping between <i>Joint Space</i> (\mathfrak{J}) and <i>Operational Space</i> (Ω) done by the robot controller.	43
Figure 2.5. General architecture of a 6R decoupled manipulator [1], and the equivalent representation (with the mechanical sense of rotation) for the KUKA KR15/2 manipulator.	45
Figure 2.6. Left, eight significant solutions for a decoupled 6R manipulator; Right, view of a KUKA KR 15/2 adopting several of these postures.	45
Figure 2.7. Vector assignment to calculate the Jacobian matrix in a standard 6R industrial manipulator (only r_1 and r_2 shown for clarity).	49
Figure 2.8. Vector assignment to calculate the Jacobian matrix in a wrist- partitioned 6R industrial manipulator, taking into account the decoupling.	50
Figure 2.9. Up, boundary singularity achieved by fully extension of the robot; down, internal singularity consequence of the alignment of q_i and q_j motion axes.	52
Figure 2.10. From left to right, <i>elbow</i> , <i>shoulder</i> and <i>wrist</i> singularities.	56
Figure 2.11 Unusable orientation workspace (or <i>degeneracy cone</i>) due to singularity in a spherical wrist [23]. This cone, exaggerated in scale, is generated by the axis of the last joint at the limit of eq. (2.31). . .	57

Figure 2.12. Highlights of the two mappings U and R, between spaces of dim=3. The axis of the sphere are oriented along the three eigenvectors of U.....	59
Figure 2.13. Best <i>orientation conditioning</i> in an orthogonal wrist, as used at many industrial wrist-partitioned manipulators.	63
Figure 2.14. Left, complete view of the KUKA robotic workcell at the IDF-UPV (Robot A, left; Robot B, right); Right, top view of the robot B synchronizaed with the rotary table.	64
Figure 2.16. Parts of the KR15/2 manipulator and main dimensions.....	65
Figure 2.17. Linear unit as an additional axis (<i>External Axis 1</i> or <i>E1</i>), with which the robot can be moved translationally (d_L).	66
Figure 2.18. Rotary (θ_M) table CR250 (<i>External Axis 2</i> or <i>E2</i>).....	67
Figure 2.19. Spatial relative position of two consecutive links and their associated coordinate frames according to the DH criterion.....	68
Figure 2.20. and Table 2.2 Frame assignments and parameters for the KUKA KR15/2 with the standard DH. The posture shown corresponds to a commonly used HOME position ($\theta_i = \{0, -\pi/2, 0, 0, \pi/2, 0\}$ in this model, for $i=1, \dots, 6$)	69
Figure 2.21. and Table 2.3. Frame assignments and parameters for the complete milling workcell at the IDF, formed by the KR15/2 manipulator mounted on the linear track and synchronized with the rotary table.	70
Figure 2.22. Workcell simulation in Matlab's Toolbox HEMERO [37].....	71
Figure 2.23. Design parameters of the workcell (h_M, d_M) and additional external joint values (θ_M, d_L). Significant position vectors in the workspace are shown.	74
Figure 2.24. The resolution by triangulation in the plane defined by $\{\theta_1, \theta_2, \theta_3\}$. Measurements in millimetres.	75
Figure 2.25. Obtaining the coordinate system $\{R''_w\}$ linked to W	77
Figure 2.26. Location of the robot flange frame on delivery.....	80
Figure 2.27. Tool holder designed for milling tasks at the IDF.	81
Figure 2.28. and Table 2.5. Robot $\{EE\}$ frame assignments and parameters for the complete DH model of the IDF workcell, with the tool holder designed for milling purposes.	82

Figure 2.29. Algorithm in Matlab to find the characteristic length (L) of a manipulator. In this particular case, table 2.2 refers to the KR 15/2 without any tool attached, giving a value of $L=350.6$ mm.	85
Figure 2.30. Left, best conditioned poture for the 6R KR 15/2 manipulator without any tool attached to the robot flange. Right, same result obtained with the RSV4W [48].	85
Figure 3.1. Repeteability and accuracy issues for a commanded target.	91
Figure 3.2. Several <i>open (up)</i> and <i>closed (down) loop</i> methods for robot calibration.	93
Figure 3.3. Generic procedure in a robot calibration using an <i>open-loop</i> method [6].	94
Figure 3.4. Generic procedure in a robot calibration using a closed-loop method [6].	94
Figure 3.5. 2D planar manipulator of two links simulating the gross positioning in the KUKA KR 15/2 manipulator.	104
Figure 3.6. 30 observed points in the workspace	106
Figure 3.7. Mechanized squared corner with high tolerance degree, placed on the base frame of the workspace $\{B\}$	112
Figure 3.8. <i>Open-loop</i> procedure proposed for the workcell calibration.	113
Figure 3.9. Left, view of the laser displacement sensor. Right, the site of the light spot on the PSD detector is dependent on the distance of the detected object. The signals A and B change depending on the position of the light spot. The calculation of the signals in the microcontroller then gives a linear output signal depending on the distance of the object.	114
Figure 3.10. Left, highlight of the DH model introduced in Chapter 2. Right, three commanded meshes on the three respective planes.	116
Figure 3.11. Coordinate $p_{S_y}^i$ approximated by the distance $D_{\Pi_y}^i$ to Π_y	117
Figure 3.12. Sweeping the reference planes with two configurations of d_L and θ_M	119
Figure 3.13. Left, the values of the increment $\Delta\beta$ show a final stable value; Right, the stop criterion is achieved after 18 th iterations.	120

Figure 4.1. The scope of CAD/CAM and CIM [2].....	127
Figure 4.2. View of the main window of NX™ while covering a complete CAD/CAM process.....	128
Figure 4.3. CAD/CAM/CNC-ROB flow process: the quality of the CAD-model determines the efficiency of the results obtained in the following steps of the manufacturing process.....	129
Figure 4.4. CAD/CAM/ROB systems offer the possibility of producing very complex pieces.....	131
Figure 4.5. Left, the path followed by PTP positioning to reach various programmed points (machining locations) on the XY axis. Right, complex contour tracking.....	132
Figure 4.6. Left, resistance welding in the industry of automotion (PTP operation). Right, continuous path (CP) tracking to cut a stone (courtesy of <i>Pedra Navas</i>).	133
Figure 4.7. Compared with a <i>linear</i> controller (left), <i>circular interpolation</i> has greatly simplified the process of programming arcs and circles, and consequently the length of the codes.	134
Figure 4.8. Input parameters prior to the generation of the toolpath (NX™)...	136
Figure 4.9. Concept of postprocessing as link between CAD/CAM and the production sytem at the shopfloor.....	137
Figure 4.10. Three different configurations in a 5-axis machine tool [14].....	139
Figure 4.11. Generic B-Y-Z-X-C milling center Huron KX8-Five, and view of the CAM software CATIA.....	140
Figure 4.12. Guo <i>et al.</i> implemented a postprocessor for the NX system (<i>Siemens Corp.</i>) and to convert CL-data to PKM control data.	140
Figure 4.13. Huang and Lin converted five-axis CL-data into robot control data readable by the unique controller of a dual-robot ABB workcell.	141
Figure 4.14. Post Builder interface that allows a simple managing of postprocessors for different standard machine-tools (within a range).	142
Figure 4.15. General structure of a CAM-ROB postprocessing.....	144
Figure 4.16. Trajectory tolerances, <i>intol</i> and <i>outol</i> , in the NX-CAM system. ..	145
Figure 4.17. NX-CAM dialog window to determine the sort of CN inputs generated. Those inputs must describe <i>linear</i> and <i>circular</i> motions for most current industrial manipulators.	146

Figure 4.18. Influence of the tolerances (<i>intol</i> and <i>outol</i>) on the number of <i>linear</i> interpolations required to track a toolpath [38].....	146
Figure 4.19. Toolpath interpolated with arcs and lines.	147
Figure 4.20. Integrated <i>postprocessing</i> in the NX-CAM system. The <i>Definition File</i> and the <i>Event Handler</i> are programmed in TCL to adapt NX's CAM to the KUKA KRC2 controller. The <i>Event Handler</i> is able to interact with executable modules programmed in C++.	148
Figure 4.21. The actual end position ($P_{\text{ACTUAL END}}$) on the arc is determined by the programmed CA sign and value, and not by the destination point ($P_{\text{PROGRAMED END}}$).....	152
Figure 4.22. Left, Arc of circumference and arc plane. Right, definition points.	153
Figure 5.1. Anatomical studies of the arm showing the movements, by Leonardo Da Vinci (1510)	160
Figure 5.2. Left, decoupled 6R manipulator. Right, the same manipulator combined with two additional joints (linear track and rotary table)	161
Figure 5.3. Irrelevant axis of symmetry of the tool at milling tasks.....	162
Figure 5.4. Intrinsic and functional redundancies of serial robotic tasks, with references to the studied workcell.	163
Figure 5.5. Milling toolpath with Frenet-Serret frames (<i>tangent t</i> , <i>normal n</i> and <i>binormal b</i>) indicating the required pose at each point in the toolpath. Again, it can be appreciated the irrelevant axis of symmetry of the milling tool.	164
Figure 5.6. Relation between the <i>desired</i> and the <i>current</i> pose.	166
Figure 5.7. Highlight of the loop leading from an initial current pose (k) to a desired final pose.	167
Figure 5.9. Additional virtual joint allowing a rotation around the symmetry axis of the tool.....	178
Figure 5.10. Decomposition of the angular velocity vector ω into two orthogonal parts: one lying on the task subspace (ω_r) and another one lying on the orthogonal task subspace ($\omega_{r\perp}$).	179
Figure 5.11. In the TDM, the motion of the secondary task is always constant in the EE frame (the rotation around the symmetry axis of EE).	181

Figure 5.12. Comparison of the DH frame assignment for the VJM (left) and the TDM (right).	182
Figure 5.13. Usual flowchart for a redundant manipulator controlling process.	183
Figure 5.14. Proposed flowchart for a redundant manipulator controlling process.	184
Figure 5.15. Block diagram of a fuzzy control. The inference mechanism interprets the values in the input vector and, based on some set of rules, assigns values to the system inputs.	186
Figure 5.16. Types of Membership Functions: triangular MF, trapezoidal MF, Gaussian MF and Sigmoidal MF.	187
Figure 5.17. Standard truth tables adapted to FL reasoning: because there is a function behind the truth table rather than just the truth table itself, values between 1 and 0 can be considered now.	188
Figure 5.18. The upper fuzzy sets (A, B) are managed with the fuzzy operations defined, to get the result displayed below.	188
Figure 5.19. Fuzzification, application of the fuzzy operator (<i>OR</i>) and implication processes for a single <i>if-then</i> rule.	190
Figure 5.20. Aggregation of the consequents across the rules a single output fuzzy set, and final defuzzification by means of the centroid method. In summary, information flows through the fuzzy inference process as shown.	191
Figure 6.1. Wrist singularity (top) and widespread position singularity (bottom) concerning the milling processes on the rotary table.	200
Figure 6.2. Overview of the Matlab's <i>Fuzzy Logic Toolbox</i> , which allows the design and testing of a fuzzy controller.	201
Figure 6.3. Flow of the heuristic reasoning in the control of the automated cell and its interaction with the expert system implemented in NX TM	203
Figure 6.4. Min-Max inference process for two rules: two discrete input values (θ_3, θ_5) are <i>fuzzified</i> (μ_3, μ_5) by the corresponding clusters involved in both rule. The minimum degree of membership in each case is taken as output membership value in the implied output clusters (red), and then aggregated (blue) into a single fuzzy set for the overall output.	205
Figure 6.5. The <i>centroid</i> calculation returns the center of area under the aggregated curve as crisp output value.	206

- Figure 6.6. Integrated postprocessing in NX. The *Definition File* and the *Event Handler* are programmed in TCL to adapt NX's CAM to the KUKA KRC2 controller. The *Event Handler* is able to interact with executable modules programmed in C++. 207
- Figure 6.7. Matlab simulation of the readjustment of the workcell after the actuation of the implemented fuzzy controller for Case A (left) and B (right). 208
- Figure 6.8. Workcell at HOME posture and main parameters of the experimental toolpath. 210
- Figure 6.9. Left, best conditioned posture for the 6R KR 15/2 manipulator deduced in Section 2.4.6. Right, mechanical mid-joint posture. . 213
- Figure 6.10. Additional virtual joint, associated with a rotation in Z_9 215
- Figure 6.11. Representation of the EE in the TDM test. The transformation matrix towards the tool tip is expressed as a displacement on Z_9 in mm. 217
- Figure 6.12. Left, graphical representation of the fuzzy engine determining the weights for the reference posture maintenance criterion. Right, the fuzzy engine giving the weights for the singularity avoidance criterion..... 227
- Figure 6.13. From left to right, representation of the peak posture of each of the three clusters in which the input spaces are divided. 228
- Figure 6.14. Output spaces for the weight assignment: left, for the reference posture criterion; right, for the singularity avoidance criterion. .. 229
- Figure 6.15. MATLAB's Fuzzy Toolbox has a Rule Editor to easily manage the *if-then* rules relating the *input* and *output* spaces. Four rules were created for the reference posture criterion (up), and two for the singularity avoidance criterion (down). 230
- Figure 6.16. Proposed Fuzzy revision for the studied workcell. 234
- Figure 6.17. Comparison of the conditioning achieved with the different VJM trials: blue, with constant weighting vector ($w=0.01$); green, with fuzzy adapted weighting vector; and red with fuzzy adapted weighting vector and a periodic revision of the IKP. 237
- Figure 6.18. While the condition number is almost the same ($k_F=0.4$), the worst posture achieved with the periodically revised method (right) has a better performance for continuous milling purposes. 238

- Figure 7.1. Up, different views of the workcell simulated in NXTM MOTION; down, two views of the simulation done with RobomoveTM. 244
- Figure 7.2. Left, real tool: an air turbine moves a 20 mm diameter spherical-tip tool; right, revolute model in NXTM. 245
- Figure 7.3. A pressurized air system (right) pushes the pistons against the opposite angle (left) to fix the workpiece. 246
- Figure 7.4. Left, EPS blanks; right, machining process of one piece in EPS.... 246
- Figure 7.5. Flow process for the cases studied 247
- Figure 7.6. Up, sketch of the scaled model (factor 1:75) of the reservoir used to simulate flows, refluxes and water retentions; down, Valencian *Falla*. 248
- Figure 7.7. The model is obtained by assembling 120 blocks of 1x1x0.5 meters of EPS, after fixing the contour lines and interpolating mesh for each block..... 249
- Figure 7.8. CAM/Rob process for the construction of each block, validating the postprocessor for 3-axis milling operations. 250
- Figure 7.9. Trajectories for Cavity Milling and Finish Milling are generated in NX..... 250
- Figure 7.10. With the programmed algorithm, the additional joints are moved to reach the complete toolpath while maintaining a well conditioned posture..... 251
- Figure 7.11. Final model in EPS with scaled factor 1:75 for flowing simulation (real dimensions of 8x13 m). 253
- Figure 7.12. *Valencian Falla* composed of fanciful *ninots* in outrageous poses arranged in a gravity-defying architecture. 254
- Figure 7.13. From left to right, the effects of the softening operation are shown. It has great relevance as it determines the rest of operations until the final milling..... 255
- Figure 7.14. For the cavity milling, the workpiece is necessarily divided in different cutting areas (upper and lower zones). This treatment optimizes the use of the additional joints, and is strongly dependent on the tool's length..... 256
- Figure 7.15. The successive cavity mill operations will be carried out with variable orientation of the tool, in order to reach all the parts of the *ninot*. 256

-
- Figure 7.16. The attachment of the blank directly over the table makes the access to the lower parts difficult. Therefore, the blank is fixed by means of an intermediate piece which raises the height..... 257
- Figure 7.17. A 5-axes toolpath is planned on the eyes to test the postprocessor. 257
- Figure 7.18. With the programmed algorithm, the additional joints are moved to obtain a better performance while maintaining a well conditioned posture..... 258
- Figure 7.19. In some regions, where the orientation of the surfaces changes rapidly, the orientation of the tool associated to them can be problematic not only for the fast reaction of the posture required but also for the collision of the tool itself. 260
- Figure 7.20. After estimating the most convenient Z_{TOOL} axis (left), the full surface could be machined almost completely (right) with a 3+2 milling operation..... 261

ABBREVIATIONS AND ACRONYMS

At each chapter, the *notation* used is introduced in its respective context. Nevertheless, common acronyms and abbreviations along the present document are listed below for shake of clarity.

AP, Accuracy of pose
{B}, Base coordinate system
CA, Circular Angle
CAD, Computer Aided Design
CAM, Computer Aided Manufacturing
CIM, Computer Integrated Manufacturing
CL-data, Cutter Location data
CNC, Computer Numerical Control *or* Computer Numerically Controlled
CP, Continuous Path
DH, Denavit-Hartenberg
DK, Direct Kinematics
DKP, Direct Kinematic Problem
DLS-inverse, Damped Least-Squares inverse
DOF, Degree of Freedom
EE, or {E}, End-Effector
FL, Fuzzy Logics
FLC, Fuzzy Logic Controller
GAs, Genetic Algorithms
GPM, Gradient Projection Method
 h , Optimized performance criterion vector or Performance vector
H, Homogeneous Jacobian matrix
HSM, High Speed Machining
IDF, Design and Manufacturing Institute – Instituto de Diseño y Fabricación
IK, Inverse Kinematics
 \mathfrak{J} , Joint Space
 J , Jacobian matrix

J_a , Analytical Jacobian
 J_g , Geometric Jacobian
KRC, KUKA Robot Controller
KRL, KUKA Robot Language
 L , Characteristic Length
MF, Membership Function
 N , links of a manipulator
NC, Numerical Control
NLSQ, Non-Linear least squares
NN, Neural Networks
PKM, Parallel Kinematic Machines
PTP, Point to Point
 r_F , Functional Redundancy
 r_I , Intrinsic Redundancy
 r_K , Kinematic Redundancy
ROB, Robotics
RP, Rapid prototyping (Repeatability of Pose, at Chapter 3)
RPY, Roll-Pitch-Yaw
RRS, Redundancy Resolution Schemes
SVD, Singular Value Decomposition
 T , Task Space
 $\{T\}$, Tool coordinate system
TCL, Tool Command Language
TCP, Tool Center Point
TDM, Twist Decomposition Method
UPV, Universidad Politécnic de Valencia
VJM, Virtual Joint Method
W, Wrist
WPI, Weighted Pseudo-Inverse
 Ω , *Operational Space*
6R, six revolute joints

CHAPTER 1
INTRODUCTION

CHAPTER 1. INTRODUCTION

1.1 INTRODUCTION

Initiated the 21st century, it is practically unbelievable the revolution experienced by the manufacturing technologies especially on last 10-15 years. The evolution of the computers, the machinery and the new communication technologies are revolutionizing the World in general, and especially the industry.

In the field of milling concerning this thesis, the revolution has already come with terms such as *High Speed Machining* (HSM) or *Rapid Prototyping* (RP), which many factories start discovering right now. Nowadays, the HSM may have some different interpretations, but it does not necessarily mean to machine with a high spindle speed. For example, some HSM applications are carried out with moderate spindle speeds (3.000-6.000 rpm) but with tools of great diameter (25-30 mm) with more global depth per cut or step-over (see Chapter 4). Clearly, the triangle material-cutter-machine conditions the cutting parameters, the milling strategies, the volume of material removed per unit of time, etc. Thus, the speeds and feeds in the process will generally depend on the material to machine. *Rapid Prototyping* in *industrial design* and in *mechanical design engineering* is of increasing importance in order to get physical replicas of CAD (*Computer Aided Design*) defined models and to support the product development process, specially when the emphasis of the design is on the surface of the product more than the replication of an inner structure. Therefore, the RP referred here is done with soft materials, such as foams.

At the same time, robotic arms are becoming more demanded in manufacturing processes involving large volumes, due to their high flexibility and large working areas. These properties are commonly increased with the use of additional joints carrying the arm or the workpiece, making up what is known as *industrial robotic workcell*. This holds in particular when the resulting prototypes are relatively large (normally, more than 0.5 metres).

In this context, conventional Computer Numerical Control (CNC) machining techniques can be adapted from being devoted to high precision metal cutting to fast milling of soft material, thus making them suited for rapid prototyping. In fact, with the implantation of more sophisticated CAD/CAM/ROB integrated manufacturing systems, the time invested in

successive verifications, adjustments and translations in the machining process up to the materialization of the product is to be reduced.

Leading commercial CAM (*Computer Aided Manufacturing*) softwares plan off-line the cutting toolpaths in a Cartesian coordinate system. Therefore, the tracking of the cutter is independent from the machine tool which will manufacture the workpiece (also due to reasons of precision and universality). These platforms are ready for the control and postprocessing of up to a maximum of *5-axis* CNC machines. These *five* parameters are, namely: three pose coordinates of the *tool center point* (TCP) and two orientations of the milling tool (considering it symmetrical along its revolute axis). It supposes no indecision in tool positioning and orientation in conventional CNC machines but, in every case, the toolpath has to be *postprocessed* (i.e., adapted) to the production system that is going to be used.

This previous overview highlights that there is still tremendous scope for improvement in the basic machine modelling and postprocessing fields. Traditional CNCs are ill-suited to the demands of many of today's complex robotic workcells. At the *Design and Manufacturing Institute*, in the *Universidad Politécnica de Valencia* (IDF-UPV), a sculpturing robot system has been configured in order to test and to apply milling methods for rapid prototyping. An industrial arm with six revolute joints is mounted on a linear track, and it works over a synchronized rotary table platform on which the initial blank of material is fixed. This provides a wider effective workspace, which is needed for handling large objects with complex shapes.

The main difficulty of postprocessing a toolpath generated by a CAM platform for a complex robotic cell focuses on the treatment to give to the redundant joints in order to avoid singularities and limits of range. With the inherent redundancy stated previously, the aim is to reach the successive poses of the toolpath in the Cartesian space. This postprocessing stage raises two differentiated tasks referring to both *cutter pose* and *manipulator posture*:

- Translation of the *cutter poses* generated by the CAM platform in agreement with the requirements of the robot language.
- Kinematic analysis of the cell for a certain requirement of the *cutter pose*, in order to include the treatment of the *robot posture* with the additional joints.

The second task arises from the fact that, with the inherent capacity to avoid non-desired postures, the set of possible configurations is now infinite. Several robot manufacturers solve the problem only by means of graphic interfaces as an intermediate step between the CAM platform and the robot execution. In these interfaces, an *expert* technician fixes the additional joints and checks the movements of the arm along the tracking, in order to know if a limit of range or a singular configuration is reached at any point. Experience and

knowledge of the technician in charge of the manufacturing process allow profiting from the employment of the additional joints in these cases. However, it is a tedious job.

This thesis focuses on the application of industrial robotic workcells to the rapid prototyping of 3D CAD-defined products. It revises diverse methods to deal with the postprocessing stage from the CAM software to the redundant workcell. It also presents an effective implementation of a CAM-ROB integrated postprocessor for a fully automatic off-line generation of the robot instructions based on both the posture and joint velocity analysis, attending to different criteria.

1.2 STATE OF ART AND CURRENT TENDENCIES

The following lines mark the current trends in the context of this thesis, on the basis of a reflection on the latter bibliographical references. Together with others, they will be recounted in the thematic area of each Chapter.

On the employment of robotic workcells for robotic rapid prototyping applications, Joe Campbell, director of strategic alliances of KUKA Robotics Corp. (Clinton Township, MI), already affirmed a few years ago [1] that “we're seeing this transition now where robots should plough being used for a lot of machining processes, in softer materials and prototyping. This is an area that was previously dominated by machine tools”. This trend has been supported in other similar analyses, among which Fei *et al.* [2] (May 2010) can be highlighted in view of its proximity to the core matter of this thesis. Nevertheless, the scope of this thesis goes beyond the merely treated by the above-mentioned authors, since it deals with other topics such as calibration and the underlying redundancies in complex workcells. Due to the multidisciplinary character of the study, the author has chosen to realize the state of the art and bibliography in relation to each Chapter.

The reader will understand that certain topics of the mechanics, robotics and classic mechatronics have been widely recorded and checked from the 50s. Some of these concepts will be raised at Chapters 2 and 3. Obviously, the sources to which the thesis will refer in those associate chapters can come from the above mentioned years though the innovation lies in its application to the framework of this thesis. For example, this is the case of the condition number of the Jacobian matrix, proposed by Jorge Angeles in the early 90s, but which still maintains its presence in the contemporary research. In this sense, a brief stay has been done in the McGill's Centre for Intelligent Machines (Montreal, Canada; www.cim.mcgill.ca), where Jorge Angeles directs his scientific research. Also in line with this, subsequent reviews of classic mechatronics applied to robot

milling can be found. It is the case of the recent publication of Xiao *et al.* [3] (January 2011) meantime the publication of this thesis.

Directly from the previous topics, the optimization in the use of mechanical complex systems has promoted numerous studies. This way, Pin *et al.* [4] (2009) also uses the condition number for the control of a robot of seven rotary joints. The profuse review documented by Chiaverini *et al.* [5] reflects the fact that classical Jacobian formulations are still in the limelight. Some researches even stem towards the empirical evaluation of different performance indexes [6]. Also referred to optimization, and later used in this thesis, the use of the fuzzy logics is a burning topic, as the recent review realized by [7] shows.

The recent studies on the optimal use of redundant robots in applications closer to milling ones are of major interest. The works of Huo *et al.* [8] about welding robots, Mitsi *et al.* [9] or Vosniakos and Matsas [10], Nemeč and Lajpah [11][10], and Olabi *et al.* [12] are the most outstanding.

As for future trends, attending to new demands that differ from the initial scope of this thesis (though it can be a point of departure) the recent publications of Neto *et al.* [13], Liu *et al.* [14] (about interfaces for the programming of industrial robots), and Sugita *et al.* [15] (about the applicability of CN generated robot toolpaths in surgery) are worth mentioning.

1.3 OBJETIVES

In 2006, the study of the CAM to robotics postprocessing with the IDF's *industrial workcell* was established as the main goal for this thesis. At the beginning, as usual in research, the final objective seemed to be clear (i.e. be able to mill with the redundant robots recently updated by KUKA with the two additional joints). Later on, with the development of the study, further partial objectives appeared as a continuous of steps.

The main objectives of this thesis are described as follows:

- Going into the knowledge of the architecture of the automated industrial redundant workcells in depth, specifically about the:
 - Establishment of a full kinematic model of the robotic workcell for both *direct* and *inverse, posture* and *velocity* analysis (Chapter 2)
 - Study of different criteria (namely, *indices*) to establish the better performance of the robot posture (Chapter 2)
 - Revision of the singularities concerning the work with this type of workcells (Chapter 2).
 - Calibration of the external joints added to the main robotic arm to form the *industrial workcell* (Chapter 3)

- Describing a complete postprocessing methodology from CAM systems to NC controllers, improving (and unifying in some cases) previous works. Those controllers manage machine tools or robots (Chapter 4).
- Giving a complete guide on the types of redundancy in industrial robots devoted to milling tasks and the *Redundancy Resolution Schemes* (RRS) associated (Chapter 5).
- Implementation and comparison of the most suitable RRS (Chapter 6).
- Raising solutions to two different applications: the milling of both an *orographic* model and a *valencian ninot*, thus having a test and evaluation of the implementation done (Chapter 7).

1.4 METHODOLOGY

As stated in the previous Section, the followed methodology arises from the partial proposed aims. In turn, these ones have appeared in correspondence with partial needs:

- The kinematic modelling of the workcell arises as direct subtask, for the need to simulate the robot and to be able to establish an off-line path planning in a PC. In short, a good model is what allows the required abstraction in the in the theoretical workframe of this thesis.
- Without losing of sight the real aim of milling, the task of calibration is carried out close to the real robot. Because of the production and assembly, the true geometric parameters of an industrial robotic workcell are different from the corresponding ones used by the robot kinematic model. It results in errors in the tool poses. Model-based robot calibration methods are studied to minimize those pose errors through identifying the true geometric parameters of the workcell based on the measurements of strategically planned toolpaths and the mathematical solutions of non-linear least squares optimization.

Those previous works were the basis to keep working with the workcell, and also to deep on the pros and cons of working with an industrial controller. Nevertheless, the final aim toward obtaining a feasible implementation was always kept in mind. At this point, the research was done in two parallel ways: on one hand, the existing CAM postprocessors and their capabilities to be reprogrammed were analyzed. In this sense, NX (licensed at the IDF) proved to be one of the most user-friendly programmable commercial platforms. To get expertise, technical advice was needed in some cases, as it is recognised in the acknowledgements Section at the beginning of the thesis. On the other hand, the mathematical models of the workcell found its uses in the Redundancy

Resolution Schemes that were to be integrated with the path tracking generated by the CAM

After all this previous work, the evaluation of the implemented models was carried out by means of a theoretical problem in 5-axis milling, i.e. the machining of an spherical surface, prior to the practice with real cases.

From the author's point of view, one of the strong points of the present document is the complete troubleshooting through the multidisciplinary approach done. Obviously, the structure of the thesis is conceived by chapters approaching each of the above mentioned matters, as it is described in the following Section,. This structure coincides with the temporary sequence of the studies.

1.5 STRUCTURE

The current thesis is planned in eight chapters, including this one.

Chapter 2, *Workcell Kinematics Characterization*, can be considered as a requirement prior to the development of the study. In summary, it consists of going into the redundant workcell architecture and the related problematic facts in depth. Therefore, this chapter also includes a state of art related to this.

In this sense, the state of art associated to each of the objectives tackled in successive chapters is made, mainly, in the first pages of each one. This also allows channelling their development.

At Chapter 3, *Workcell calibration*, a *Non-Linear Least Squares* (NLSQ) identification model has been derived from the consistency conditions of three orthogonal a pattern planes that are swept by a laser displacement sensor mounted on the manipulator. This non-contact calibration scheme can be implemented autonomously. It is expected to be suitable for on-site calibration in an industrial environment of the external joints introduced in Chapter 2.

Chapter 4, *CAM to Workcell postprocessing*, lays the necessary foundations for the knowledge of integrated production systems. In this chapter, the fundamentals of CAM systems and CNC are described, to finally raise the concept of *postprocessing*. This chapter ends with particular specifications related to the system NX-CAM and the KUKA workcell used in the IDF.

On the scope of the thesis, due to the fact that the workcell is redundant, Chapter 5 compiles and classifies different types of redundancy and describes the different methods for redundancy resolution (namely, Redundancy Resolution Schemes, RRS) that can be considered for postprocessing labours.

From the previous chapters, several implementations for the particular IDF's redundant workcell are done in Chapter 6. The first implementation

consists of a postprocessor based on the analysis with logic fuzzy of the inverse kinematics (IK) of the robotic arm posture. This allows getting on in postprocessor programming labours. Nevertheless, with the limitations found for complex millings (large workpieces or milling with variable tool orientation), the acquired practice is then invested in the implementation of the RRS based on the control at joint rate level.

Chapter 7 applies the postprocessor implemented to two practical cases: the machining of an orographic surface of big dimensions, and the one of a *valencian ninot* with variable tool orientation.

Finally, the most relevant conclusions are outlined in Chapter 8. A series of feasible future works departing from the current investigation is also proposed.

REFERENCES (Ch. 1)

- [1] Waurzyniak P., Shop-Floor Productivity, *Manufacturing Engineering* July 2005 Vol. 135 No. 1
- [2] Fei M., Haiou Z., Guilan W.; Application of industrial robot in rapid prototype manufacturing technology, *IEEE 2nd International Conference on Industrial Mechatronics and Automation – ICIMA 2010*, pp. 218-220, Wuhan, China, May 30-31, 2010.
- [3] W. Xiao, H. Strauß, T. Loohß, H-W Hoffmeister, J. Hesselbach; Closed-form inverse kinematics of 6R milling robot with singularity avoidance, *Prod. Eng. Res. Devel.* (2011) 5:103–110
- [4] W. Pin, W. Lili, L. Hong, D. Qian; An Optimization Algorithm for Redundant 7R Robot, *2009 First International Workshop on Education Technology and Computer Science - IWETCS, China, 2009.*
- [5] S. Chiaverini, G. Oriolo, I. D. Walker; Kinematically redundant manipulators; *Springer Handbook of Robotics*, B. Siciliano and O. Khatib (Eds.), Springer-Verlag, Berlin, D, pp. 245–268, 2008.
- [6] M. Tisius, M. Pryor, C. Kapoor, D. Tesar, An Empirical Approach to Performance Criteria for Manipulation, *Journal of Mechanisms and Robotics*, AUGUST 2009, Vol. 1 / 031002-1/12
- [7] R-E. Precup, H. Hellendoorn; A Surrey on industrial applications of fuzzy control; *Comput.Industry* (2010), doi:10.1016/j.compind.2010.10.001 (in press)
- [8] L. Huo and L. Baron, The joint-limits and singularity avoidance in robotic welding; *Industrial Robot: An International Journal*, Volume 35, Number 5, 2008, pp. 456–464
- [9] S. Mitsi, K.-D. Bouzakis, D. Sigris, G. Mansour, Determination of optimum robot base location considering discrete end-effector positions by means of hybrid genetic algorithm, *Robotics and Computer-Integrated Manufacturing* 24 (2008) 50–59
- [10] G.-C. Vosniakos, E. Matsas; Improving feasibility of robotic milling through robot placement optimisation, *Robotics and Computer-Integrated Manufacturing* 26 (2010) 517–525
- [11] B. Nemeč and L.Z. Lajpah; Robotic cell for custom finishing operations, *International Journal of Computer Integrated Manufacturing*, Vol. 21, No. 1, January – February 2008, 33 – 42
- [12] A.Olabi, R. Bearee, O. Gibaru, M. Damak, Feedrate planning for machining with industrial six-axis robots; *Control Engineering Practice* 18 (2010) 471–482
- [13] P. Neto, J. N. Pires, A. P. Moreira; CAD-Based Off-Line Robot Programming; *2010 IEEE Conference on Robotics, Automation and Mechatronics*
- [14] Z. Liu, W. Bu, J. Tan; Motion navigation for arc welding robots based on feature mapping in a simulation environment; *Robotics and Computer-Integrated Manufacturing* 26 (2010) 137–144
- [15] N. Sugita, T. Nakano, T. Kato, Y. Nakajima and M. Mitsuishi; Path Generator for Bone Machining in Minimally Invasive Orthopedic Surgery; *IEEE/ASME TRANSACTIONS ON MECHATRONICS*, VOL. 15, NO. 3, JUNE 2010, pp. 471-479

CHAPTER 2

WORKCELL KINEMATIC CHARACTERIZATION

*“And them the breaks / For
we designer fakes / We need
to concentrate / On more
than meets the eye” –
20 years (Placebo)*

CHAPTER 2. WORKCELL KINEMATIC CHARACTERIZATION

2.1. CONCEPTS ON MANIPULATOR KINEMATICS

A manipulator is a device that helps human beings to perform manipulating tasks. A robotic manipulator is to be distinguished from the previous for its ability to lead itself through computer control. Once programmed, it can implement the same task repeatedly. In general, robotic manipulators can be studied using the concept of kinematic chain. A kinematic chain is a set of rigid bodies, also called links, coupled by kinematic pairs.

A kinematic pairs is the coupling of two rigid bodies so as to constrain their relative motion. There are two basic types of kinematic pairs, namely, *upper* and *lower kinematic pairs*. An upper kinematic pair is obtained through either line contact or point contact, and thus, appears in cam-and-follower, gear trains, and roller bearings, for example. A lower kinematic pair occurs when contact takes place along a surface common to the two bodies. From the six common lower kinematic pairs (planar, spherical, cylindrical, revolute, prismatic, and helicoidal) [10][11], *prismatic* and *revolute* are the most employed in industrial manipulators (both allowing only one *degree of freedom*, DOF), Figure 2.1.

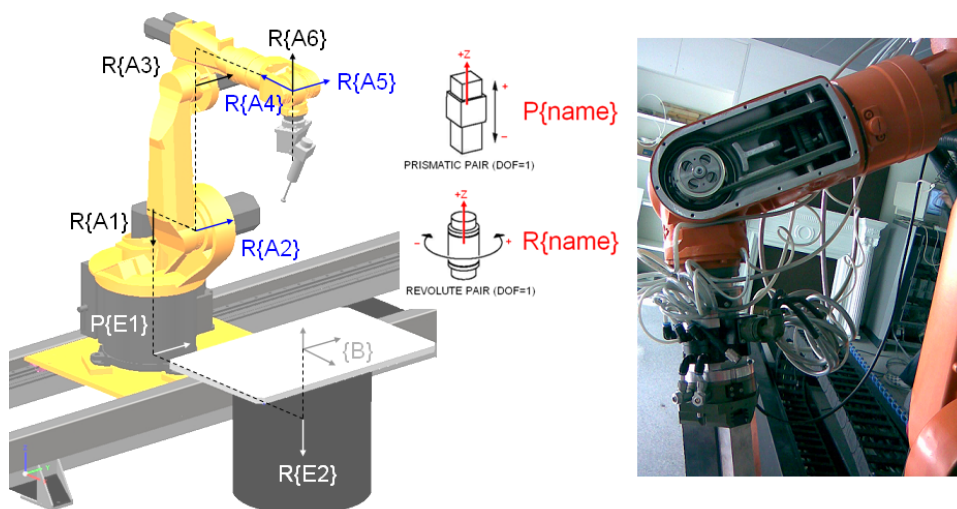


Figure 2.1 Left, Prismatic (P) and Revolute (R) joints at an industrial workcell. Right, detail of the construction of a revolute joint .

2.1.1. Joint variables (generalized coordinates)

This thesis focuses on serial manipulators, i.e., simple open kinematic chains. In such manipulators, there are exactly two bodies with a degree of connectivity¹ of one, called end-bodies, and all the other bodies with a degree of connectivity of two. One end-body is arbitrary regarded as fixed and is named the *Base* {B}, while the other end-body is regarded as movable and is called the moving body, or the *end-effector* (EE) of the manipulator, Figure 2.2.

A total of $6N$ coordinates are required to specify the position and orientation of all the N links of a manipulator relative to a coordinate frame (namely, the *posture* of the manipulator). Since the links are coupled together, the $6N$ coordinates can be expressed as functions of a minimum set, $q \in R^n$.

$$q = (q_1, q_2, q_3, \dots, q_n)^T \in \mathfrak{S}^n; \quad n = \dim(\mathfrak{S}) \quad (2.1)$$

The q joint variables of the manipulator, that are all independent, are known as *generalized coordinates*, and the motions associated with them are consistent with the constraints. The value n is the *degree of freedom* (DOF) for that system, and is the sum of DOF of each joint. We will refer \mathfrak{S} to as the *joint space*, whose dimension is n ; and *general n -axis manipulator* to as any serial robot having such a dimension.

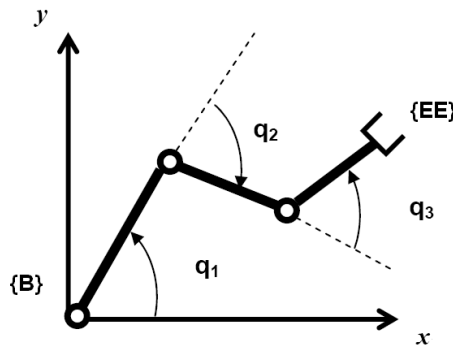


Figure 2.2. Generalized coordinates in a planar manipulator

¹ The *degree of connectivity* of a body is defined as the number of bodies directly connected to the said body through kinematic pairs.

2.1.2. Operational coordinates

The operational coordinates of a robot are the m components of the vector $x \in R^m$ that specifies the position and the orientation (namely, *pose*) of the EE of the robot in the physical space (namely, *operational space*, Ω) with regard to an operational frame of reference (*Base*, $\{B\}$), generally Cartesian:

$$x = (x_1, x_2, x_3, \dots, x_m)^T \quad (2.2)$$

In case of the general movement of the terminal organ in the 3D space, depending on the type of coordinates of orientation that are in use, m might be major than six. Nevertheless, since it is preferable that above mentioned coordinates are independent, m generally will be equal to six. In such a case, three coordinates define the position of a point of the body (TCP or *tool center point*), whereas other three define the orientation angles around that point regarding one notation conventionalism, usually Roll-Pitch-Yaw (RPY) or whatever of the different Euler notations².

$$x = (p_x, p_y, p_z, \theta_x, \theta_y, \theta_z) \in \Omega; \quad \dim(\Omega) = m = 6 \quad (2.3)$$

Nevertheless, in the mathematical background of the kinematic analysis of manipulators, it is also useful the *homogeneous* notation, in which the position and orientation of a coordinate system (usually the EE) referred to another (usually B), Figure 2.3, is expressed by means of a 4x4 matrix, ${}^B T_E$, namely:

$${}^B T_E = \begin{bmatrix} \mathbf{i}_x & \mathbf{j}_x & \mathbf{k}_x & p_x \\ \mathbf{i}_y & \mathbf{j}_y & \mathbf{k}_y & p_y \\ \mathbf{i}_z & \mathbf{j}_z & \mathbf{k}_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

where the last column indicates the position of origin of $\{E\}$, O_E , with respect to $\{B\}$, and the first three columns are the coordinates of the unitary vectors defining $\{E\}$ projected onto $\{B\}$ (see Figure 2.3).

Specifically, when assuming a parallel-jaw gripper as the terminal organ of the robot, $\{\vec{i}, \vec{j}, \vec{k}\}$ will be referred to as $\{\vec{n}, \vec{s}, \vec{a}\}$, regarding the expected

² Due to the existing confusion found in Euler Angle notations, we will adopt the the *KUKA KRC2* controller convention. Thus RPY values are defined as three consecutive rotations in Z, Y and X axes, respectively, over the resulting moved axis after each rotation. [46] [http://en.wikipedia.org/wiki/Euler_angles]

motions in those respective directions *normal*, *sliding* (or *open-close*) and *approach*, Figure 2.3.

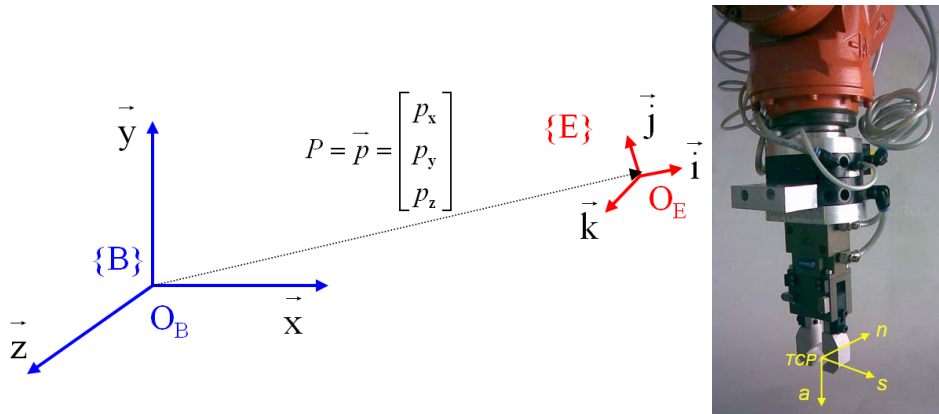


Figure 2.3. Left, coordinate system {E} referred to {B} by means of ${}^B T_E$. Right, conventionalism in the specific case of the terminal organ of the robot.

It is notable that in certain particular cases, the movement of the EE might not happen in the 3D space. In fact, for example, a flat movement of the terminal organ might be sufficient for some tasks. In such a case, the number of operational coordinates can diminish to 2 or to 3, depending on if the orientation is relevant or not for the task. This fact leads to what is named *functional redundancy* that will be further tackled in Chapter 5.

2.2. LEVEL OF KINEMATIC ANALYSIS

To adequately control the position and orientation the robot during a task, *kinematic models* are required to establish the mathematics description of the mechanical systems. This kinematic analysis can be raised from three perspectives [1]:

- The relations between joint positions and Cartesian positions of the EE, known as *displacement analysis*;
- The relations between the time-rates of change of the joint positions, (joint rates), and the rate of the EE. This is known as *velocity analysis*;
- The relations between the second time-derivatives of the joint positions, referred to as the joint accelerations, with the time-rate of change of the twist of the EE, known as *acceleration analysis*.

In the context of this thesis, only the *displacement* and *velocity* analysis will be considered as means of control an *industrial* manipulator at postprocessing milling tasks. There are two main reasons for that decision: the first one comes from the typical closed architecture of industrial manipulators (also from the practical point of view), only allowing the control by position parameters and velocity parameters within a range. The second reason is the type of work aimed to do, i.e., prototyping in soft materials at the velocities perfectly assumed by this category of robots (normally working at the 10% of the maximum possible velocity at pick and place tasks).

2.2.1. Direct and Inverse Kinematic Problem at the displacement level.

Figure 2.4 represents the mapping between joint space and operational space at the displacement level. The *Direct Kinematic Problem* (DKP) is the mapping from *Joint Space* (\mathfrak{J}) to *Operational Space* (Ω), i.e., determining the pose of the EE (position and orientation) for a given manipulator in a given posture. On the contrary, the *Inverse Kinematic Problem* (IKP) is the mapping from Ω to \mathfrak{J} , determining the posture of a given manipulator for a given pose of its EE.

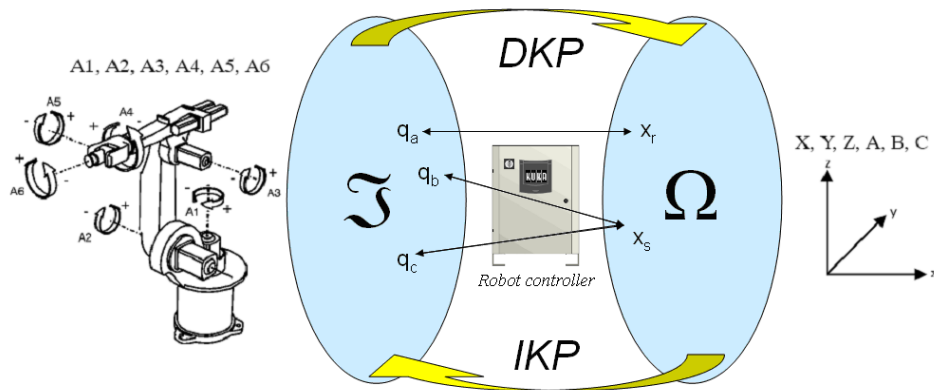


Figure 2.4. Mapping between *Joint Space* (\mathfrak{J}) and *Operational Space* (Ω) done by the robot controller.

The DKP can be written as a nonlinear algebraic system,

$$x = DKP(q) = f(q) \quad (2.5)$$

where q is a point in \mathfrak{S} and x the corresponding point in Ω . The function $DKP(\cdot)$ allows the computation of the operational space variables x from the knowledge of the joint space variables q . For the *straightforward*³ problem stated in (2.5), the Denavit-Hartenberg (DH) model [1][34] is employed in this thesis due to its simplicity and popularity in the robotics community⁴ (see Section 2.4.2.).

Alternatively, the IKP is also written as a nonlinear algebraic system of the form

$$q = IKP(x) = f^{-1}(x) \quad (2.6)$$

At the displacement level, the DKP is straightforward and admits a single solution, i.e., a point in \mathfrak{S} represents a unique pose of the EE in Ω . In general, the IKP is much more complex and challenging since it requires the solution of a highly non-linear algebraic system, for which no analytical closed-form solution exist for a general 6R manipulator. Several or even infinite number of solutions may exist (in the case of a redundant manipulator, see Chapter 5). Thus, some suppositions must be done to discriminate a valid solution. In these cases, also the mechanical joint limits of real robots may reduce the number of reachable solutions.

Several methods have been described to analytically solve the IK by means of numerical or graphical methods, and they are revised in [1]. Instead, many manipulators in industry have three last succeeding revolute joints with their axes intersecting at a point (W), as shown in Figure 2.5. Pieper [8] showed that a 6R manipulator, termed as *decoupled manipulator* or *wrist-partitioned*, always has closed-form solutions.

Tsai and Morgan [13] found that, although the number of real-significant solutions changes from case to case, the total number of significant solutions (real and complex) for all the 6R manipulators is 16, but it is reduced to 8 significant solutions for the decoupled cases (Figure 2.6).

³ Note that the DKP solution may be computed for any manipulator, irrespective of the number of joints or kinematic structure. All these calculations can be easily programmed, being common the use of the *Robotics Toolbox* for *Matlab* [36] or *Hemero* [37], as shown in Figure 2.22.

⁴ Two differing methodologies have been established for assigning coordinate frames, resulting in a standard-DH notation [1][34] and a modified-DH (MDH) notation [35]. However, in many studies this differentiation is not noted leading to some confusion.

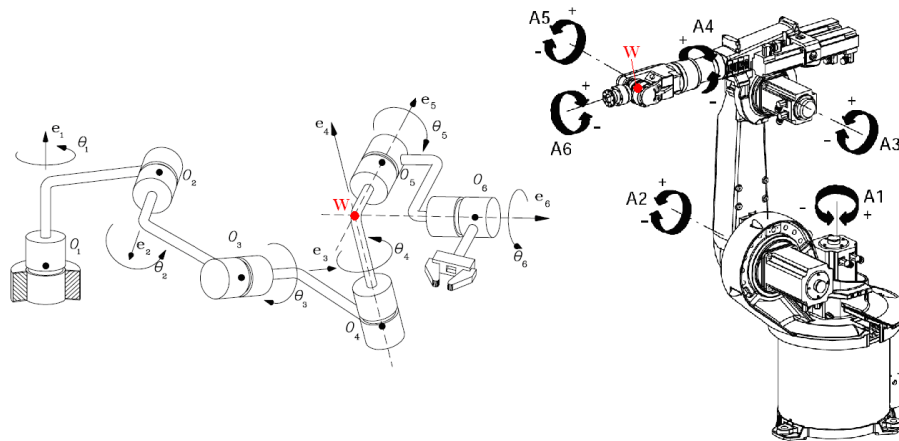


Figure 2.5. General architecture of a 6R decoupled manipulator [1], and the equivalent representation (with the mechanical sense of rotation) for the KUKA KR15/2 manipulator.

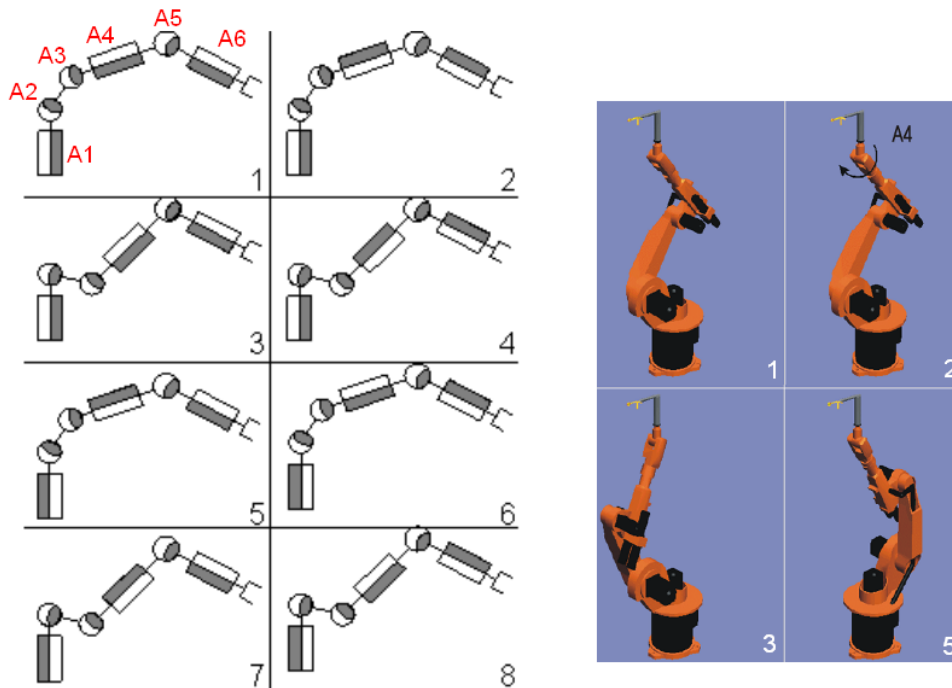


Figure 2.6. Left, eight significant solutions for a decoupled 6R manipulator; Right, view of a KUKA KR 15/2 adopting several of these postures.

2.2.2. Kinematic analysis at joint-rate level.

Differential kinematics of robot manipulators was first introduced by Whitney [9]. He proposed to use differential relationships to solve the joint space motion from a given Cartesian space motion of the EE, namely, the *resolved-motion rate* control.

i) *DKP at joint-rate level.*

The relationship between the EE velocity and the joint velocity is represented by a linear algebraic equation, namely

$$t = J \cdot \dot{q} \quad (2.7)$$

Equation (2.7) states the DKP at joint-rate level, or *forward kinematics problem*. The coefficient of the linear equation is the *Jacobian* matrix (J), which is a non-linear function of joint angles. This matrix maps the joint rates, grouped into the n -dimensional vector $\dot{q} = [\dot{q}_1, \dots, \dot{q}_n]^T$, into the EE velocity, represented as the m -dimensional twist array t , or *twist* vector [1], namely

$$t = \begin{bmatrix} \omega \\ v \end{bmatrix} \quad (2.8)$$

with ω and v denoting the angular and linear velocities of the EE reference frame relative to the fixed base frame $\{B\}$, respectively.

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}; \quad v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \quad (2.9)$$

ii) *IKP at joint-rate level*

Equation (2.7) implies that if the joint velocities \dot{q} are known, then the twist of the EE can be obtained. Most often, the inverse problem is required, i.e. given the desired twist of the EE the aim is to obtain the joint velocities, with J known.

In the case of non-redundant robots, J is a square matrix; hence, the solution can be found as

$$\dot{q} = J^{-1} \cdot t \quad (2.10)$$

From a practical point of view, this approach is evaluated numerically for a given posture of the robot since the symbolic handle of J is cumbersome due to the trigonometric entries of this matrix [11]. Actually, J^{-1} does not need to be calculated explicitly if the LU-decomposition method [2] is used to solve the system of equations [1].

As the value of J changes with the movement of the robot, at certain postures it may not have inverse (being the $\det(J)=0$). Those postures, namely *singular configurations*, will be treated later in this Chapter.

Another difficulty in solving the inverse problem arises in the case of robots having a value of $m < n$, in the sense of (2.1) and (2.3). From a practical point of view, those manipulators (described later as *redundants*, see Chapter 5) have a not-square matrix J (with more columns than rows). Thus, the system of equations is underdetermined having infinite possible solutions.

The solution \dot{q} that better fits all the equations of the system (2.10) with a minimum least squares criterion can be achieved with the use of the right Moore-Penrose pseudo-inverse⁵ (J^\dagger)

$$J^\dagger = J^T (JJ^T)^{-1} \quad (2.11)$$

$$\dot{q} = J^\dagger \cdot t \quad (2.12)$$

Equation (2.12) minimizes the Euclidean norm of the residual, $\|J \cdot \dot{q} - t\|_2$ that brings $J \cdot \dot{q}$ “as close as possible” to t . In short, (2.12) results in a *minimum-norm* solution. It has been broadly used at the velocity level to minimize $\|\dot{q}\|_2$, which can be viewed as a minimization of energy consumption [12].

⁵ Given an $m \times n$ matrix B , the Moore-Penrose generalized matrix inverse is a unique $n \times m$ matrix pseudoinverse B^\dagger . The Moore-Penrose inverse satisfies

$$BB^\dagger B = B; \quad B^\dagger BB^\dagger = B^\dagger; \quad (BB^\dagger)^T = BB^\dagger; \quad (B^\dagger B)^T = B^\dagger B$$

It is also true that $z = B^\dagger \cdot c$ is the shortest length least squares solution to the problem $Bz = c$

Nevertheless, a *homogeneous component* can be added to (2.12) in order to optimize a *secondary task* with an additional criterion (at the cost of giving up the minimum-norm solution). Thus, this general non-minimum-norm solution can be written as:

$$\dot{q} = \overbrace{J^\dagger t}^{\text{Minimum-norm solution}} + \overbrace{(I - J^\dagger J)h}^{\text{Homogeneous solution}} \quad (2.13)$$

In this case, some other criteria can be applied, which usually consider a second task to be performed by the robot. These methods fall into what is named *Redundancy Resolution Schemes*, and they will be analysed at Chapter 5.

iii) *The Jacobian matrix*

It is noticeable that there are two different conceptions for the Jacobian matrix (J), namely, the *geometric* and the *analytical* Jacobian. Mainly, they differ on the method for expressing the rotation velocity of the operation point [11][14][16].

The *analytical* Jacobian (J_a) can be obtained by differentiation of the m functions $\{f_1, \dots, f_m\}$ of the DKP of position, eq. (2.5), that is,

$$J_a(q) = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \dots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \dots & \vdots \\ \frac{\partial f_m}{\partial q_1} & \dots & \frac{\partial f_m}{\partial q_n} \end{bmatrix} = \frac{\partial f}{\partial q} \in R^{6 \times n} \quad (2.14)$$

However, it is computationally cumbersome to try to evaluate the analytic Jacobian matrix.

In 1972, Whitney [15] proposed the *geometric* Jacobian (J_g) matrix to simplify the computation⁶. For the sake of brevity, brief but well-known indications to obtain J_g are extracted from [1]: in summary, the J_g matrix of a general n-axis manipulator has the form

⁶ For sake of brevity, we will refer J_g as J indistinctly

$$J_g = [j_1 \ j_2 \ \dots \ j_n] \quad (2.15)$$

For revolute (R) and prismatic (P) joints, the 6-dimensional i^{th} column of $J_g, j_i (i = 1, \dots, n)$, is given as

$$R: j_i = \begin{bmatrix} e_i \\ e_i \times r_i \end{bmatrix}; \quad P: j_i = \begin{bmatrix} 0 \\ e_i \end{bmatrix} \quad (2.16)$$

where e_i is the unit vector parallel to the axis of the i^{th} revolute joint, and r_i is the vector from any point on that axis to the considered operational point (in the EE), as shown in Figure 2.7.

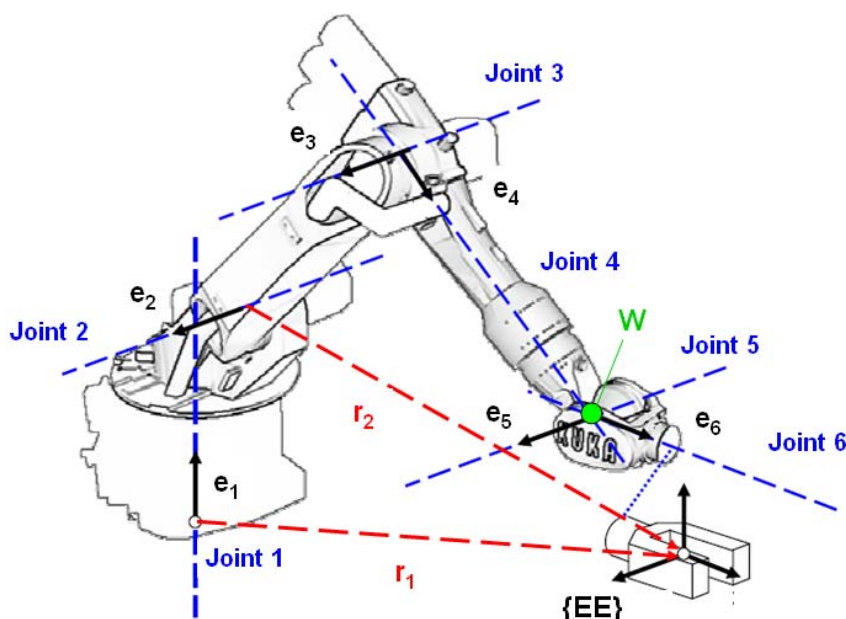


Figure 2.7. Vector assignment to calculate the Jacobian matrix in a standard 6R industrial manipulator (only r_1 and r_2 shown for clarity).

- **Consideration to wrist-partitioned manipulator Jacobian**

In the *decoupled* manipulators introduced at the end of the section 2.2.1. , the positioning and orienting problems can be considered separately. In fact for many tasks, if W is the EE reference point, arbitrary displacements can be assumed as the translation of point W combined with

the orientation of the EE reference frame, whose origin is W. Indeed, the wrist is also named *spherical* because, when W is fixed, then all points on the wrist move on spheres centred at W.

As the determinant of the Jacobian of a six-axis robot is invariant under a change of the EE reference point [1][14]. In some cases can be useful the consideration of W as this point. By following the method previously described, we note that the location of W in the base reference frame is independent of last three joint angles. In the most common case of a 6R decoupled manipulator we have:

$$v_W = \dot{q}_1 e_1 \times r_1 + \dot{q}_2 e_2 \times r_2 + \dot{q}_3 e_3 \times r_3 \quad (2.17)$$

r_i being the position vector of W with regard to any point on the first three axes, and e_i the direction vector of the axes, both expressed in coordinates of the base frame, Figure 2.8.

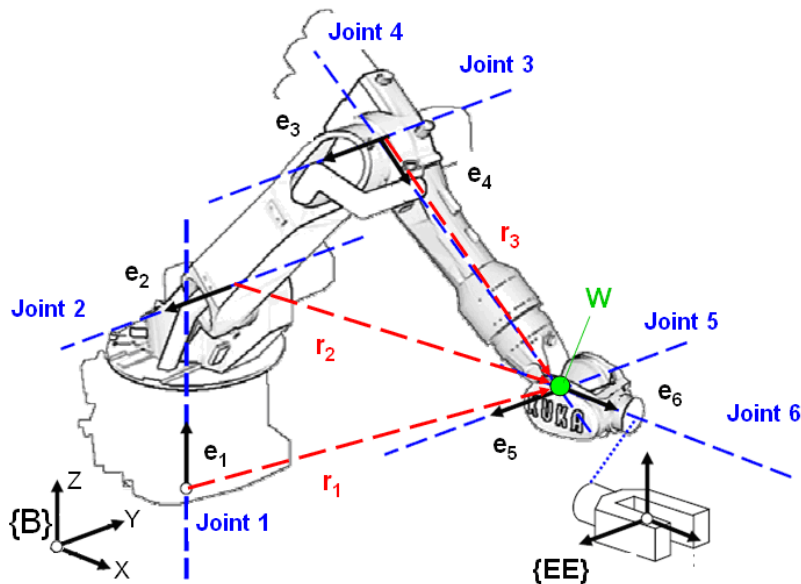


Figure 2.8. Vector assignment to calculate the Jacobian matrix in a wrist-partitioned 6R industrial manipulator, taking into account the decoupling.

The angular velocity vector, ω , of the EE reference frame whose origin is on C can be written as the vector sum of the contributions of the angular velocities of the individual joints:

$$\omega = \omega_1 + \omega_2 + \dots + \omega_6 = \dot{q}_1 e_1 + \dot{q}_2 e_2 + \dots + \dot{q}_6 e_6 \quad (2.18)$$

Finally, Jacobian takes the form

$$J = \begin{bmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ e_1 \times r & e_2 \times r & e_3 \times r & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & 0_{3 \times 3} \end{bmatrix} \quad (2.19)$$

From (2.7), the problem stated to these robots can be resumed as

$$\begin{bmatrix} \omega \\ v_W \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & 0_{3 \times 3} \end{bmatrix} \begin{bmatrix} \dot{q}_{pos} \\ \dot{q}_{ori} \end{bmatrix} \quad (2.20)$$

where \dot{q} has been separated into \dot{q}_{pos} and \dot{q}_{ori} to denote the three-dimensional vectors of *arm* and *wrist* joint rates, respectively.

Thus, the velocity inversion of this type of manipulators can be done by means of:

$$\begin{aligned} \dot{q}_{pos} &= J_{21}^{-1} v_W \\ \dot{q}_{ori} &= J_{12}^{-1} (\omega - J_{11} \dot{q}_{pos}) \end{aligned} \quad (2.21)$$

2.2.3. Singular configurations.

The *singular configurations* of a manipulator are those postures in which the geometric Jacobian matrix becomes rank-deficient. By the fact, when J is rank-deficient the mobility of the kinematic chain is reduced, i.e., at least one of the possible motions of the EE in Ω disappears.

In the case of non-redundant manipulators (with square Jacobian), the determinant of J is zero. It is remarkable that J has not inverse at those postures, and infinite solutions to the IKP may exist. Similarly, when computing the active joint velocities with the pseudoinverse of J in a redundant serial-link manipulator, the singularity arises when J loses its full rank.

From a computational point of view this implies that the system cannot be solved for \dot{q} , and the control of the robot becomes problematic. In the neighbourhood of a singularity, a small variation in the Cartesian movement of the EE may cause large velocities in the joints. Actually, J raises its condition number, which causes great imprecision when solving (2.10). This aspect will be tackled later in this Chapter (see Section 2.3.2.)

Singularities can be classified in two categories (Figure 2.9):

- Boundary singularities: when the manipulator is outstretched or retracted. They are easily avoided on condition that the robot is not working near the limits of its reachable workspace.
- Internal singularities: they occur inside the reachable workspace. Generally they are consequence of the alignment of two or more motion axes. These singularities constitute a serious problem in many off-line planned operations, as many milling tasks which are in the scope of this thesis.

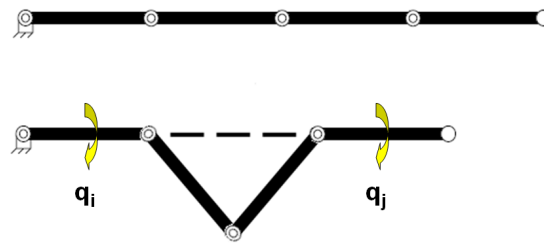


Figure 2.9. Up, boundary singularity achieved by fully extension of the robot; down, internal singularity consequence of the alignment of q_i and q_j motion axes.

The possibility that a manipulator adopts an internal singular configuration during the execution of a task was raised in one of the first works about kinematic modelling of manipulators [15]. Later on, many authors have dealt with the characterization of the singular configurations of manipulators [18][17], and others have also considered the prevention of these configurations and the better conditioning number of the Jacobian matrix as criteria for the design of manipulators [1][19][20][21][22]. Due to its relevance, several implications of the singularity configurations will be revised in Section 2.3.

i) Consideration to wrist-partitioned manipulator singularities

For the sake of this thesis, special attention is given to decoupled manipulators, whose singularities have already been a major research area.

In practice, all *industrial* models get blocked near a singular configuration to avoid a possible damage of the internal mechanisms. Despite this fact, operating manuals give either an insignificant treatment of this subject, or none at all [30]. Most users of 6R robots are only

acquainted with the operating manuals of their specific robot, but not of the specific literature, which generally requires an advanced level of mathematical and geometric knowledge.

Hayes et al. [18] made a revision of the concept for such manipulators, but also giving a geometric interpretation of how the singularities arise, given the structure of the associated Jacobian. From (2.19), it is clear that

$$\det(J) = -\det(J_{12}) \cdot \det(J_{21}) \quad (2.22)$$

Moreover, many of these manipulators follow a common structure with the first axis constantly pointing along the Z-axis of the base frame $\{B\}$, and both axes 2 and 3 parallel to each other and to the XY-plane of $\{B\}$, see Figure 2.8. Thus, (2.19) can be expressed as

$$J = \begin{bmatrix} 0 & e_{2x} & e_{2x} & e_{4x} & e_{5x} & e_{6x} \\ 0 & e_{2y} & e_{2y} & e_{4y} & e_{5y} & e_{6y} \\ e_{1z} & 0 & 0 & e_{4z} & e_{5z} & e_{6z} \\ -e_{1z}r_{1y} & e_{2y}r_{2z} & e_{2y}r_{3z} & 0 & 0 & 0 \\ e_{1z}r_{1x} & -e_{2x}r_{2z} & -e_{2x}r_{3z} & 0 & 0 & 0 \\ 0 & e_{2x}r_{2y} - e_{2y}r_{2x} & e_{2x}r_{3y} - e_{2y}r_{3x} & 0 & 0 & 0 \end{bmatrix} \quad (2.23)$$

and (2.31) is reduced to calculate

$$\begin{aligned} \det(J) &= A \cdot B \cdot C \\ A &\equiv (r_{2z}e_{2y}r_{3x} - e_{2x}r_{2z}r_{3y} + e_{2x}r_{3z}r_{2y} - r_{3z}e_{2y}r_{2x}) \\ B &\equiv (r_{1y}e_{2x} - e_{2y}r_{1x}) \\ C &\equiv (e_{4x}e_{5z}e_{6y} - e_{4x}e_{6z}e_{5y} + e_{4y}e_{5x}e_{6z} - e_{4y}e_{6x}e_{5z} + e_{4z}e_{6x}e_{5y} - e_{4z}e_{5x}e_{6y}) \end{aligned} \quad (2.24)$$

By analysing the conditions that voids each factor, we get the well known following singularities:

i.a) Elbow singularity

Without loss of generality, the robot can be considered at the posture in which e_2 is parallel to the YZ plane of $\{B\}$. In this case,

$$e_2 = [0 \quad 1 \quad 0]^T \quad (2.25)$$

Thus, equation A vanishes when

$$\begin{aligned} r_{2z}r_{3x} - r_{3z}r_{2x} = 0 &\rightarrow r_{2z}/r_{3z} = r_{2x}/r_{3x} \rightarrow \\ &\rightarrow \frac{|r_2|\cos(q_2)}{|r_3|\cos(q_3)} = \frac{|r_2|\sin(q_2)}{|r_3|\sin(q_3)} \end{aligned} \quad (2.26)$$

In general it is satisfied whenever r_2 and r_3 are aligned, but considering the joint limits and interference, elbow singularity is therefore restricted to satisfy $q_3 = \pm q_2$.

For the scope of this thesis, we note that the elbow singularity surface represents the limits of the workspace (previously termed as *boundary singularities*). Clearly, elbow singular configurations can be easily anticipated and avoided by keeping the EE at a safe distance from its limits.

i.b) Shoulder singularity

Vanishing of the factor B means

$$r_{1y}e_{2x} - e_{2y}r_{1x} = 0 \quad (2.27)$$

If e_{2x} or e_{2y} vanishes (and being aware of the fact that, in those cases, $e_{2y} = 1$ or $e_{2x} = 1$, respectively) then B will vanish only if $r_{1x} = 0$ or $r_{1y} = 0$, respectively.

It means that point W lies in the YZ-plane, or ZX-plane respectively, of the Base frame $\{B\}$. Because of the construction of common 6R manipulators, W is consequently supposed to be on the Z-axis in this plane. If neither e_{2x} nor e_{2y} vanishes, then (2.27) must accomplish $r_{1x} = 0$ and $r_{1y} = 0$. Again, it forces W to be on the Z-axis of the Base frame (see Figure 2.10).

i.c) Wrist singularity

Looking at C in eq. (2.24), we note that the vanishing of this depends on the relative orientation of the last three axes (4, 5 and 6). Without loss of generality, we can consider axis 4 to be fixed relative to the others and the base frame, for example.

$$e_4 = [1 \ 0 \ 0]^T \quad (2.28)$$

Thus, the condition to satisfy, $C=0$, is reduced to

$$e_{5z}e_{6y} - e_{6z}e_{5y} = 0 \quad (2.29)$$

Because of the construction of the wrist, axes 4 and 5 as well as axes 5 and 6 are always perpendicular. Again, without loss of generality, we can suppose axis 5 satisfying one of these two cases (namely, (a) and (b)), both perpendicular to axis 4, and the consequences for (2.29) in each case

$$\begin{aligned} e_{5(a)} &= [0 \ 1 \ 0]^T \Rightarrow 0 \cdot e_{6y} - e_{6z} \cdot 1 = 0 \rightarrow e_{6z} = 0 \\ e_{5(b)} &= [0 \ 0 \ 1]^T \Rightarrow 1 \cdot e_{6y} - e_{6z} \cdot 0 = 0 \rightarrow e_{6y} = 0 \end{aligned} \quad (2.30)$$

As stated before, because of the construction of the wrist, we note that factor C vanishes in every case by taking $e_6 = e_4$.

As a conclusion, the condition for wrist singular configurations is only satisfied when axes 4 and 6 are parallel (see Figure 2.10).

For the scope of this thesis, we note that this condition can be satisfied in the entire reachable workspace (i.e., is an *internal singularity*), being the one most problematic in milling tasks consisting of a path tracking generated by a CAM system.

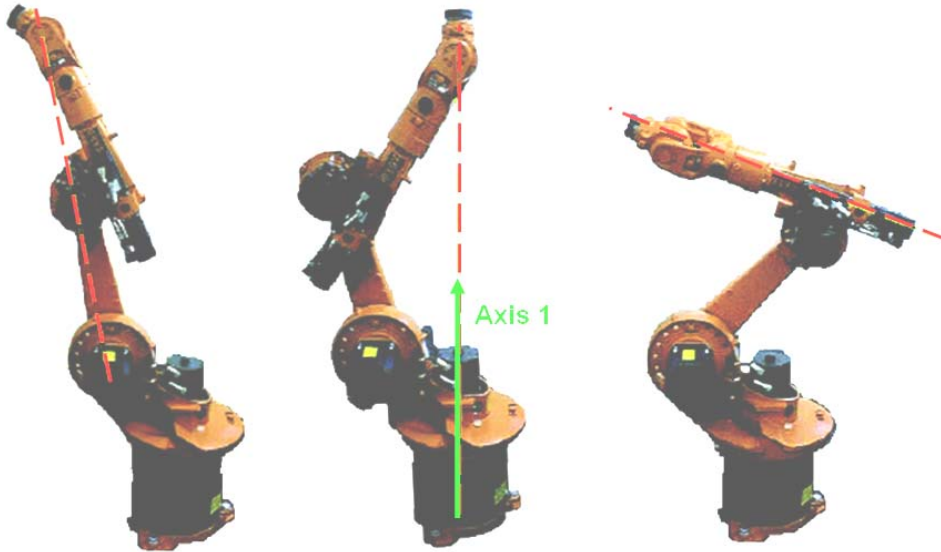


Figure 2.10. From left to right, *elbow*, *shoulder* and *wrist* singularities.

2.3. KINETOSTATIC PERFORMANCE INDICES. POSTURE-DEPENDENT INDICES.

The fundamental purpose of the kinematic chain of a robotic manipulator consists of driving the EE in the workspace with an efficient controlled movement that allows carrying out a task. As stated in the previous section, there exist certain configurations of these architectures that originate a *poor performance* of the robot. Tasks carried out near these configurations can mean, for example, an excessive operation of the actuators, the inability to realize locally some movements, or a low precision in the positioning of the terminal tool. Paul and Stevenson [23] named such configurations as *degenerated*.

It seems to be desirable to have a characterization of the kinematic performance of a manipulator, which can help to typify the configurations that, in contrast with the degenerated ones, originate an optimal running of the robot. Angeles [1] defines the *kinetostatic performance index* of a robotic mechanical system (*kinetostatic index* for brevity), as “a scalar quantity that measures how well the system behaves with regard to force and motion transmission, the latter being understood in the differential sense, i.e., at the velocity level”.

It is enormous the relevance of these indices in the field of robotic design, but also as a criterion for robot control. For the sake of this thesis, in the following section we revise some *posture-dependent indices* that will be considered as a performance criterion for the calculation of the ideal emplacement of a manipulator on a redundant workcell, in order to perform a

task. We focus the discussion below to only two indices, namely, *manipulability* and *kinetostatic conditioning index*.

2.3.1. Manipulability

Although several authors [24][25][26] made first approaches in the topic of the *kinematic performance*, the concept of *manipulability* was introduced by T. Yoshikawa [27]. Previously, Paul and Stevenson [23] had used the absolute value of the determinant of the Jacobian to measure the kinematic performance of spherical wrists. They termed as degenerated any configuration that approach a value of zero passing a threshold, namely

$$\psi = |\det(J)|; \quad \psi \leq 0.5 \quad (2.31)$$

although no clear justification is presented for using 0.5 in this relationship.

In the vicinity of this condition, they observed that the terminal organ is very poorly sensitive to the joint motions. In contrast, they noticed that a manipulator using only configurations with high values of ψ worked in an efficient way. Figure 2.11 highlights the idea of two degeneracy cones corresponding to a spherical wrist (only the cone affecting the operational configurations within the mechanical limits is depicted). They also noticed that the maximum range of possible work of the spherical wrist is got with its three axes mutually orthogonal.

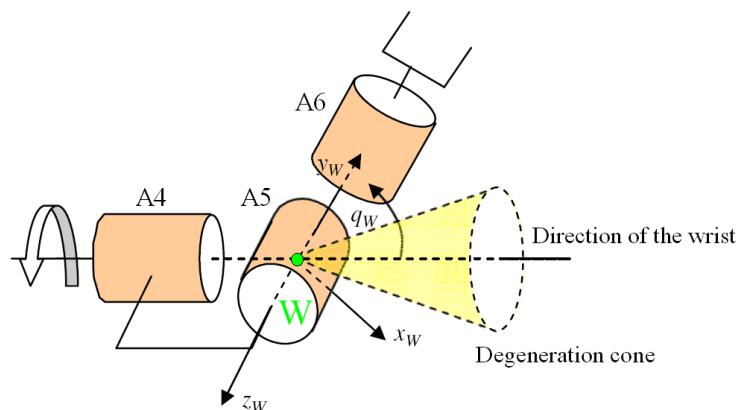


Figure 2.11 Unusable orientation workspace (or *degeneracy cone*) due to singularity in a spherical wrist [23]. This cone, exaggerated in scale, is generated by the axis of the last joint at the limit of eq. (2.31).

Yoshikawa [27] defined the *manipulability* as the square root of the determinant of the product of the manipulator Jacobian by its transpose.

$$\mu = \sqrt{\det(JJ^T)} \quad (2.32)$$

It can be noted that, for a square Jacobian, (2.32) is identical to the absolute value of the determinant of the Jacobian and hence it coincides with Paul and Stevenson's performance index.

It is possible to get a geometric interpretation of the concept of manipulability (Figure 2.12). From the fact that (2.7) maps \dot{q} into t , we can factor J into an orthogonal matrix (R) and a positive-semidefinite⁷ matrix (U), by invoking the *polar-decomposition theorem*, namely

$$J \equiv RU \quad (2.33)$$

Thus, we can interpret a concatenation of two mappings of the form

$$t = J \cdot \dot{q} \rightarrow \begin{cases} y = U\dot{q} \\ t = Ry \end{cases} \quad (2.34)$$

- The U matrix maps the unit m -dimensional ball into an m -axis ellipsoid (whose semiaxis lengths bear the ratios of the eigenvalues of U , which are the same that those of J).
- The matrix R maps this ellipsoid into another one with identical semiaxes, but rotated or reflected, depending upon whether R is proper or improper orthogonal⁸ (Figure 2.12).

For evaluation purposes, the Jacobian of a serial manipulator can be studied as mapping the unit ball in the space of joint rates, that is

$$\|\dot{q}\| = (\dot{q}_1^2 + \dot{q}_2^2 + \dots + \dot{q}_n^2)^{1/2} \leq 1 \quad (2.35)$$

into a rotated or reflected ellipsoid in the space of Cartesian velocities, known as *manipulability ellipsoid*.

⁷ A positive semidefinite matrix is a Hermitian matrix all of whose eigenvalues are non-negative. (<http://mathworld.wolfram.com/PositiveSemidefiniteMatrix.html>)

⁸ Improper rotations can be represented by orthogonal matrices with determinants of -1 .

In this context, the value of the manipulability (μ) is the product of the eigenvalues (σ) of J (or U) and is proportional to the volume of the ellipsoid in the space of Cartesian velocities.

$$\mu = \sqrt{\det(JJ^T)} = \sqrt{\det(UU^T)} = \sigma_1 \cdot \sigma_2 \cdot \dots \cdot \sigma_m \quad (2.36)$$

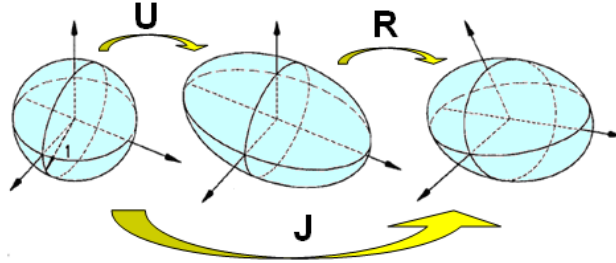


Figure 2.12. Highlights of the two mappings U and R , between spaces of $\dim=3$. The axis of the sphere are oriented along the three eigenvectors of U .

Hence, it also can be observed as a measure of the efficiency of the transformation of the articulate speeds into the terminal organ velocities in the different directions of the operational space. If J is singular, then at least one semiaxis vanishes and the ellipsoid degenerates into a disk (without volume). Manipulators at singular configurations thus have a manipulability of zero.

Finally, if the kinematic chain of a manipulator contains a part that essentially contributes to the positioning of the EE, and another segment that deals basically with its orientation, then the manipulability calculated by means of (2.32) theoretically would allow to evaluate globally the efficiency with which the translation displacements take place so much as those of rotation of the EE. Nevertheless, due to the lack of homogeneity of J , this index turns out to be inappropriate to evaluate in an effective way both types of displacement. Considering this, in a later work [28] Yoshikawa proposed two complementary indices: one to quantify the translational manipulability and other one to evaluate the rotational manipulability. Both are calculated applying (2.32), but for the case of the translational manipulability it uses the sub-matrix of J_{21} of eq. (2.19), proceeding in an analogous way for the rotational manipulability. These indices of translational and rotational manipulabilities only apply in case of decoupled robots.

2.3.2. Condition number of J

The manipulability previously introduced points to evaluate the kinematic invertibility of J at (2.10). However, Angeles [1][22] observed that in some circumstances the determinant of a matrix is meaningless in assessing the invertibility of that matrix. Chiaverini [32] remarks that the manipulability measure may remain constant even in the presence of significant variations of either the condition number or the smallest singular value of J . As a consequence, a kinetostatic index should not be founded on the determinant of J (or on the determinant of the product JJ^T).

In the numerical analysis, the *condition number* associated with a problem is a measure of its adequacy to digital computation, that is, how numerically well-conditioned the problem is⁹. In particular, the condition number of J , $k(J)$, can be considered as the rate at which the solution of (2.7), \dot{q} , will change with respect to a small change in t , but also gives an upper bound for the roundoff-error amplification in the solution, by (2.37). Then, it is noteworthy that the notion of *matrix conditioning* also can be used to estimate the *kinematic sensitivity* [3] in resolution of (2.7).

$$\frac{\|\delta\dot{q}\|}{\|\dot{q}\|} \leq C(J(q)) \frac{\|\delta t\|}{\|t\|}; \quad \text{with } \begin{cases} \delta\dot{q}, \text{ error of the joint rates} \\ \delta t, \text{ error of the rates in the EE} \end{cases} \quad (2.37)$$

Thus, if the condition number is large, even a small error in t may cause a large error in \dot{q} , and the problem is said to be *ill-conditioned*. On the other hand, if the condition number is small then the error in \dot{q} will not be much bigger than the error in t and the problem is said to be *well-conditioned*.

As a conclusion, it is desirable for the robot to work at any postures minimizing $k(J)$, as it determines the robustness against manufacturing, assembly, or joint-encoder errors [19].

i) Inhomogeneity of J , and Characteristic length

As introduced in the previous section, in many cases the definition of the condition number also has to deal with the lack of homogeneity of J . That is the case of manipulators for both positioning and orientation tasks,

⁹ Note that this is before any effect of round-off error is taken into account, i.e. conditioning is a property of the matrix J , not the algorithm or accuracy of the computer used to solve (2.7).

in which for every column of J , eq. (2.16)-(R), the first three entries are dimensionless while the last three have units of length.

In order to avoid this dimensional inhomogeneity, given the descriptive geometric parameters of a robot (the DH model introduced in Section 2.2.1.) we can evaluate the *characteristic length* (L) of the robot [19][29] by which we divide the Jacobian entries that have units of length (2.38). In this way we get a homogeneous Jacobian (H), whose associated condition number becomes meaningful.

$$H = [h_1 \ h_2 \ \dots \ h_n]; \quad h_i = \begin{bmatrix} e_i \\ \frac{1}{L} e_i \times r_i \end{bmatrix} \quad (2.38)$$

The characteristic length is defined as “the normalizing length that renders the condition number of the Jacobian matrix a minimum” [1]. For a given manipulator by its DH parameters, L and hence its minimum condition number can be obtained by an optimization method described in [19] as *direct problem*.

ii) Formula for the condition number of J

The condition number of a dimensionally homogeneous Jacobian (H) is

$$k(H) = \|H\| \cdot \|H^{-1}\| \quad (2.39)$$

where $\|\cdot\|$ represents any matrix norm. Due to its lower computational cost¹⁰ compared with other widely used norms, Angeles [1] suggests adopting a weighted¹¹ Frobenius norm, namely

$$\|H\|_F \equiv \sqrt{\frac{1}{m} \text{tr}(H^T H)} = \sqrt{\frac{1}{m} \text{tr}(HH^T)} \quad (2.40)$$

¹⁰ The computation of k_F requires only the inversion of a positive-definite 6x6 matrix. On the contrary, the computation of k_2 , with the matrix 2-norm, requires an iterative procedure to calculate the eigenvalues of HH^T . [22]

¹¹ In the weighting factor considered, m refers to the dimension of the operational space, which is 3 for positioning tasks and 6 for positioning and orientating tasks (as the applications to be discussed in this Thesis)

This norm is invariant under isometric transformations (reflections or rotations) of the m -dimensional operational space. It means that a frame-invariant condition number will be obtained.

As stated previously, for non-redundant manipulators H is square, and hence the condition number is

$$k_F(H) = \|H\|_F \|H^{-1}\|_F = \frac{1}{6} \sqrt{\text{tr}(HH^T) \cdot \text{tr}[(HH^T)^{-1}]} \quad (2.41)$$

However, for redundant manipulators H is rectangular. Nevertheless, the calculus can be tackled with its *right pseudo-inverse* (H^\dagger) as defined in (2.11). Thus, (2.39) applies with H^\dagger instead H^{-1} . It is easy to prove that applying (2.40) we get

$$\|H^\dagger\|_F = \|H^T (HH^T)^{-1}\|_F = \sqrt{\frac{1}{6} \text{tr}[(HH^T)^{-1}]} \quad (2.42)$$

Then, it is concluded that (2.41) is an expression valid for any $6 \times n$ matrix, with $n \geq 6$.

In spite of the norm used, the value of the condition number will be comprised from one to infinity ($1 \leq k < \infty$). It attains the value of unity for matrices with non-zero identical eigenvalues (which we saw that map the unit ball into another ball, Figure 2.12). Such matrices are called *isotropic*. As a consequence, Angeles [1] terms as *isotropic manipulator* all those whose Jacobian matrix is isotropic at certain postures (thus, induce the smallest possible roundoff-errors). On the other side, singular matrices have a singular value that vanishes, and then their condition number is infinite.

iii) *Consideration to wrist-partitioned manipulator singularities*

From (2.21), we observe that the accuracy of the computed joint rates depends only on blocks J_{12} and J_{21} . As J_{12} accounts for the orientation of the EE, it seems to be logical to call the conditioning associated this submatrix as the *orientation conditioning*. Analogously, J_{21} accounts for the positioning of W (Figure 2.8) and so, the conditioning associated with this submatrix is termed the *position conditioning* [22].

Angeles and Rojas [31] (also Angeles in [1]), shown that an orthogonal wrist such as the described in section 2.2.3. (i.e. first and second axes of the wrist as well as second and third axes of the wrist being

always perpendicular) attains an *orientation conditioning* of unity if the joint between the last two links is at a right angle. In this case, the wrist is postured so that its three axes are mutually orthogonal¹², as shown in Figure 2.13.

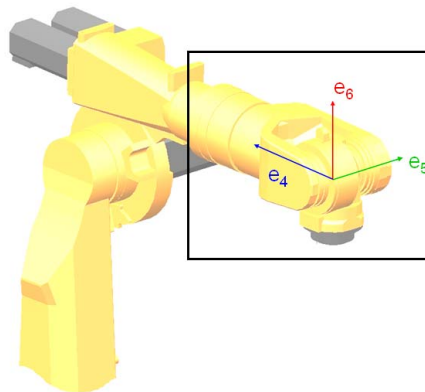


Figure 2.13. Best *orientation conditioning* in an orthogonal wrist, as used at many industrial wrist-partitioned manipulators.

It is remarkable that the calculation of the condition number of spherical wrists is straightforward, as all entries of J_{12} are dimensionless. In the case of commonly 6R decoupled manipulators, all the entries of J_{21} associated with the three-revolute positioning axes have units of length and hence calculation of the condition number is also straightforward. As stated in section 2.3.2. -i), problems arise when considering general six-axis manipulators for both positioning and orientation tasks, dealing with the homogeneization of J by means of the characteristic length, L .

Finally, Angeles [22] shown that wrist-partitioned manipulators having isotropic arm and wrist subchains do not have an overall isotropic Jacobian matrix.

2.4. KINEMATIC CHARACTERIZATION OF AN INDUSTRIAL WORKCELL.

As this thesis focuses in integrating a CAM system with a robotic workcell, further transformations will require knowledge of the architecture of

¹² As not all the partitioned wrists can attain a condition of unity, but the success of the orthogonal wrist in industry is an example of mechanical desing instinct leading to an optimal desing (since these wrist were introduced in the robot market before the condition number and the isotropy became a criteria)

the workcell in order to solve the mappings previously introduced in Section 2.2. The following characterization will be distinguished for the robotic workcell placed at the *Intituto de Diseño y Fabricación (IDF)*, since it is the employed in next Chapters to study the postprocessing of milling toolpaths.

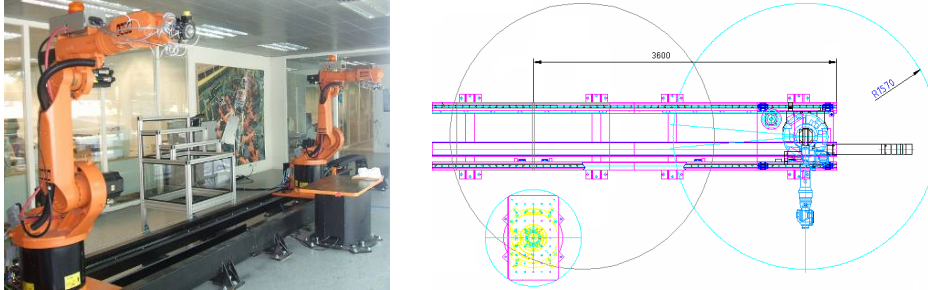


Figure 2.14. Left, complete view of the KUKA robotic workcell at the IDF-UPV (Robot A, left; Robot B, right); Right, top view of the robot B synchronized with the rotary table.

2.4.1. Components of the numerically controlled KUKA workcell

The workcell being studied at the IDF consists of a 6R KUKA KR15/2 manipulator synchronized with a rotary table on which milling operations will be carried out, in addition to the linear track on which it is mounted (Figure 2.14).

Both additional joints, with the other six rotary joints of the main robotic chain, complete a workcell with eight degrees of movement.

The exposed configuration provides a high degree of flexibility in milling works due to its large working areas and multiple possible configurations obtained with the additional joints.

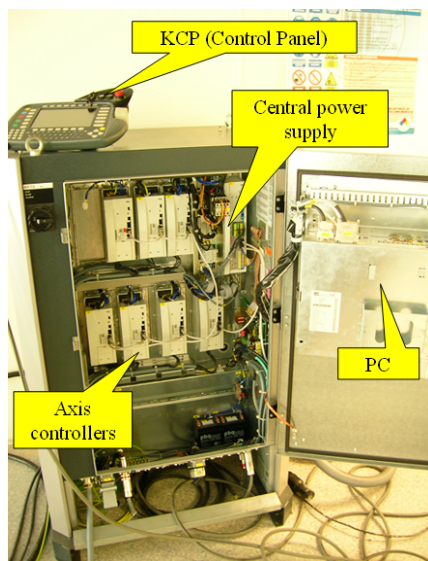


Figure 2.15. *KUKA KRC2 controller*

i) KUKA KRC2 controller

It contains all the components and functions which are required to operate the robot. It comprises the computer and power units, which are both installed in a common control

cabinet [33] (see Figure 2.15).

ii) KUKA KR15/2 manipulator

It is an *industrial* decoupled 6R manipulator widely used for pick-and-place, assembly or welding tasks (see Figure 2.16). This model, of 235 kg weight, is characterized for its repeatability ($< \pm 0.1$ mm) and velocity (up to 2 m/s) for low payloads (up to 15 kg). The robot consists of a base frame, on which the rotating column turns about a vertical axis together with the link arm, arm and wrist. The wrist is provided with a mounting flange for the attachment of the EE (e.g. grippers, welding or milling tools).

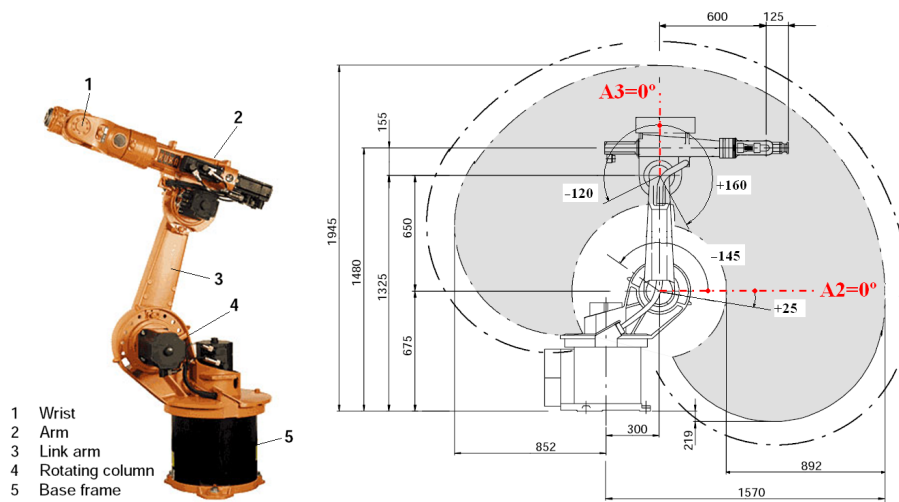


Figure 2.16. Parts of the KR15/2 manipulator and main dimensions.

The rotational axes and directions of rotation in motion of the robot are depicted in Figure 2.5, within the ranges specified at Table 2.1. and Figure 2.16.

<i>Axis</i>	<i>Range of motion</i>	<i>Max. speed</i>
1	$\pm 185^\circ$	152 °/s
2	-145° to $+25^\circ$	152 °/s
3	-120° to $+160^\circ$	152 °/s
4	$\pm 350^\circ$	284 °/s
5	$\pm 135^\circ$	293 °/s
6	$\pm 350^\circ$	604 °/s

Table 2.1. *Range of motion* referred to the mechanical zero of the robot axis concerned. It is limited by means of software switches on all joints.

iii) *Additional linear axis*

The KL 250 is a self-contained one-axis linear unit. It is operated as the *External Axis 1* (E1) of the robot and is controlled by the *KRC2* control cabinet. For sake of brevity, in this thesis this joint and the value of its motion are indistinctively designated as d_L . The main components are depicted in Figure 2.17.

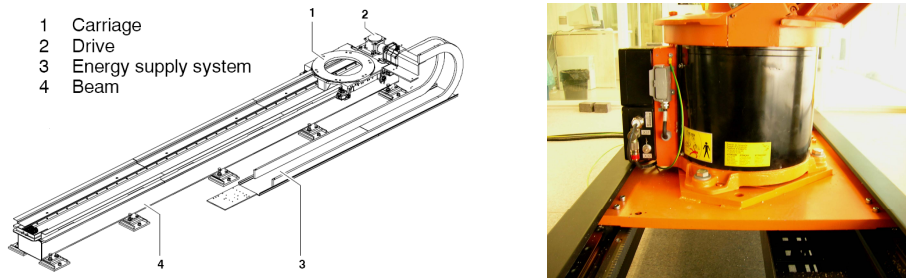


Figure 2.17. Linear unit as an additional axis (*External Axis 1* or *E1*), with which the robot can be moved translationally (d_L).

The movement range is restricted by programmable software limit switches and is additionally safeguarded by mechanical stops if these limit switches are overrun.

iv) *Additional rotary table*

The CR250 rotary table is formed by a base frame in which an AC servomotor drives the operating surface. It is operated as the *External Axis 2* (E2) of the robot and, like *E1*, it is also controlled by the *KRC2* control cabinet. For sake of brevity, in this thesis this joint and the value of its motion are indistinctively designated as θ_M .

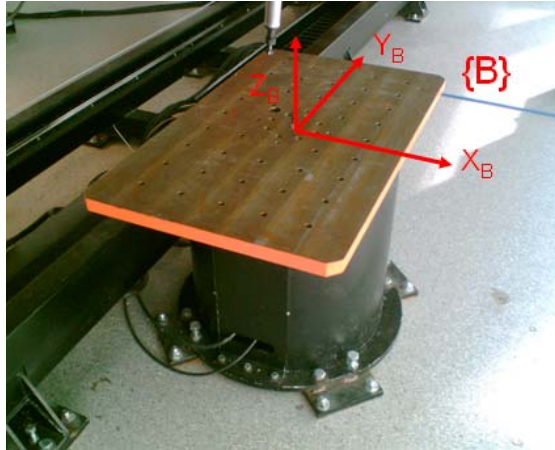


Figure 2.18. Rotary (θ_M) table CR250 (External Axis 2 or E2).

2.4.2. Direct Kinematic model of position

The DH model previously introduced in Section 2.2.1. is represented as a 4×4 matrix T that results from the product:

$$T = \prod_1^n {}^{i-1}A_i \quad (2.43)$$

T defines the transformation of the n -link associated coordinate frame into the coordinate frame associated to the base $\{B\}$ of the robotic arm. A_i designates the DH transformation matrix relating frame i to frame $i-1$. The n^{th} link frame coincides with the EE's coordinate frame.

Figure 2.19 illustrates the spatial relative position of two consecutive links and their associated coordinate frames. The DH model adopts four parameters ($a_i, \alpha_i, d_i, \theta_i$) to describe the transformation, including translations and rotations, from one link ($i-1$) to the next (i). The first parameter, a_i , represents the length of common normal of the two link axes. The second parameter, α_i , denotes the angle between the two link axes. The remaining two parameters describe the relative position of two adjacent links, which are provided by their distance d_i and their rotation angle θ_i .

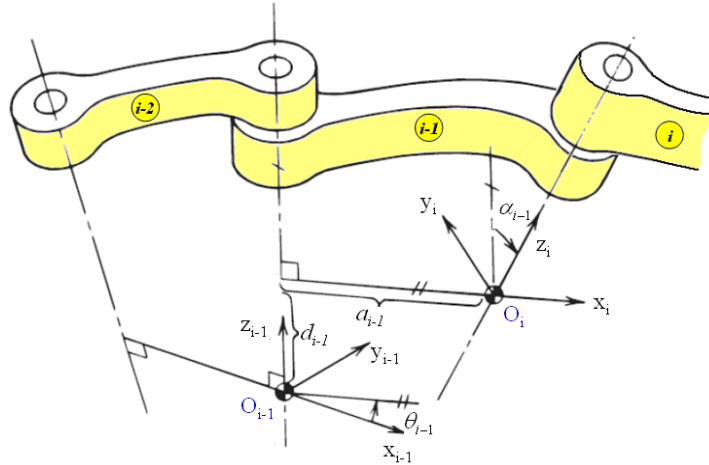


Figure 2.19. Spatial relative position of two consecutive links and their associated coordinate frames according to the DH criterion.

After the DH coordinate frame is constructed for each link, the transformation from one link to the next is described by the following homogeneous matrix:

$$\begin{aligned}
 {}^{i-1}A_i &= \text{Rot}(Z_i, \theta_i) \text{Trans}(Z_i, d_i) \text{Trans}(X_{i+1}, a_i) \text{Rot}(X_{i+1}, \alpha_i) = \\
 &= \begin{bmatrix} \cos \theta_i & -\cos \alpha_i \sin \theta_i & \sin \alpha_i \sin \theta_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \alpha_i \cos \theta_i & -\sin \alpha_i \cos \theta_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.44)
 \end{aligned}$$

For a revolute axis θ_i is the joint variable and d_i is constant, while for a prismatic joint d_i is variable, and θ_i is constant. In particular, all the parameters describing the KUKA KR15/2 are summarized as follows:

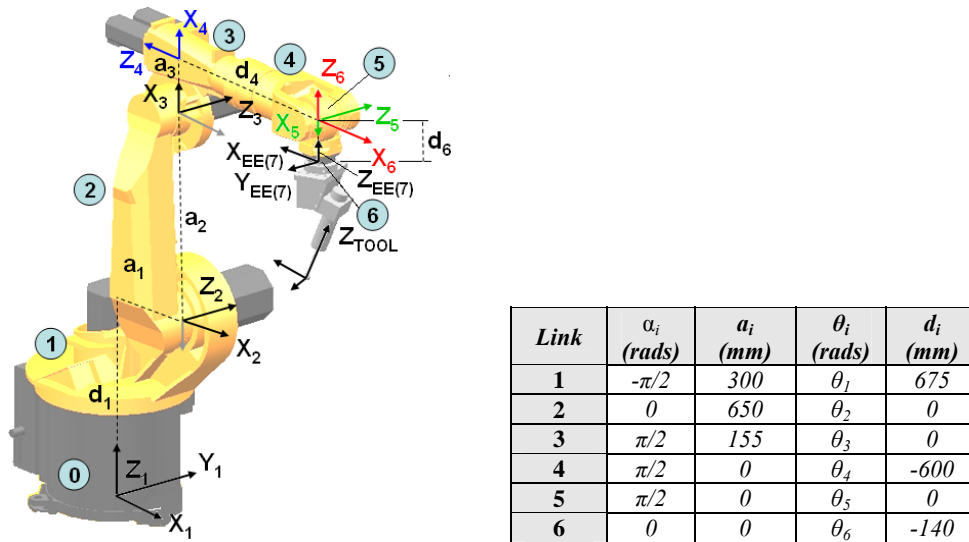
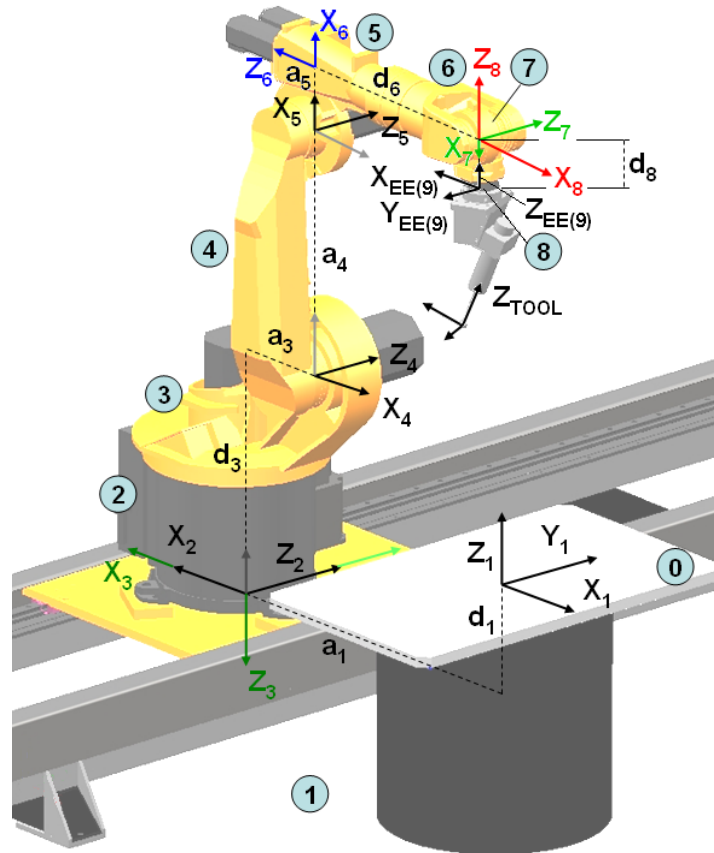


Figure 2.20. and Table 2.2 Frame assignments and parameters for the KUKA KR15/2 with the standard DH. The posture shown corresponds to a commonly used HOME¹³ position ($\theta_i = \{0, -\pi/2, 0, 0, \pi/2, 0\}$ in this model, for $i=1, \dots, 6$)

In the complete workcell studied, it was done the assignment shown in Figure 2.21. In this case, the complete workcell can be assumed as rotating around the workpiece coordinate system placed on the working space (table). This consideration is going to simplify further calculations from a pure kinematic perspective. The posture shown corresponds to a commonly used HOME position (in this model, $\theta_i = \{ \pi, \pi, -\pi/2, 0, 0, \pi/2, 0 \}$ for $i=M, 1, 2, \dots, 6$; and $d_L=0$)

¹³ The HOME position is a well known posture of the robot, previous to any task to be done.



Link	α_i (rads)	a_i (mm)	θ_i (rads)	d_i (mm)
1	$\pi/2$	803	θ_M	-305
2	$\pi/2$	0	0	d_L
3	$\pi/2$	300	θ_1	-675
4	0	650	θ_2	0
5	$\pi/2$	155	θ_3	0
6	$\pi/2$	0	θ_4	-600
7	$\pi/2$	0	θ_5	0
8	0	0	θ_6	-140

Figure 2.21. and Table 2.3. Frame assignments and parameters for the complete milling workcell at the IDF, formed by the KR15/2 manipulator mounted on the linear track and synchronized with the rotary table.

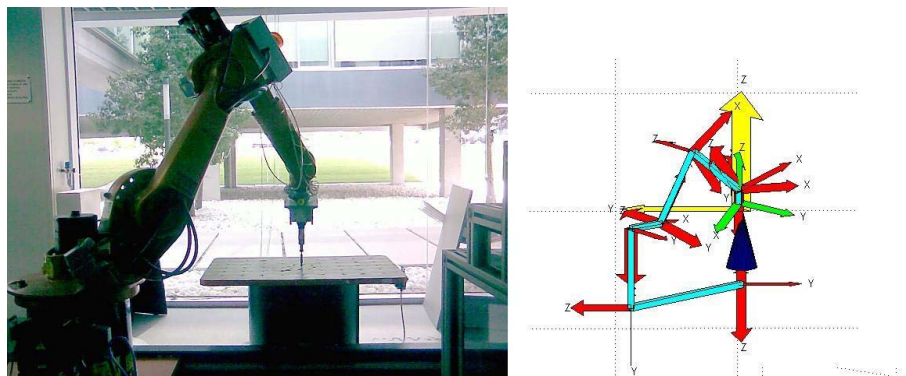


Figure 2.22. Workcell simulation in Matlab's Toolbox HEMERO [37].

2.4.3. Inverse Kinematic Problem (IKP) of position

The IKP is interesting since, in practice, the task specifications and desired toolpaths in industrial applications are defined in the Cartesian Operational Space Ω , while the robot controller work at the Joint Space \mathfrak{J} . In the particular case of this research, CAM systems generate the tracking of the TCP in the Cartesian space for reasons of *universality* of this kind of data (prior to adaptation to any capable machine). This data is also related directly with the desired finish conditions which are mandatory in any milling task.

As previously stated, the KR15/2 is a *decoupled* manipulator (Figure 2.5). For the KR15/2 (and other rotary robot arms), various arm configurations can be defined according to human arm geometry and the link coordinate systems [38]. Thus, we can assume that this manipulator has eight significant solutions to the IKP (Figure 2.6). There are four solutions for the first three joints: two for the right shoulder arm configuration (Figure 2.6, 1-4) and two for the left shoulder arm configuration (Figure 2.6, 5-8). For each arm configuration, there are two sets of joint solutions to the last three joints of Figure 2.20: $(\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\})$ and $(\{\theta_1, \theta_2, \theta_3, \theta_4+\pi, -\theta_5, \theta_6+\pi\})$ [40].

In general, IKP can be solved either by an *algebraic*, *iterative*, or *geometric* approach [41]. A brief discussion can be outlined about the better method:

- The *algebraic* approach suffers from the fact that the solution does not give a clear indication of how to select the correct solution from the several possible solutions for a particular arm configuration [38][42][43].
- The *iterative* solution often requires more computations and it does not guarantee convergence to the correct solution, especially in the singular

and degenerate cases. Furthermore, there is no indication of how to choose the correct solution for a particular arm configuration [38].

- If the manipulator under consideration is simple, that is, it is a *decoupled* model, then the *geometric* approach presents a better approach to get a closed-form solution [38][39].

The existence of mechanical joint limits reduces the number of reachable solutions for the given manipulator. Actually, due to the workcell distribution and mainly the kind of work carried out on the rotary table, the manipulator studied in this Section has little multiplicity when solving the IKP: according to the previous definitions, only the *right and above arm* solutions will be taken into account when solving the gross positioning (Figure 2.6, 1-2)¹⁴.

i) Geometric approach for the IKP of position

To practical effects, this Section will introduce a *geometric* approach to get a consistent joint angle solution of the KR15/2 manipulator for a desired pose of the EE. At this point, the full workcell has *infinite* solutions due to its *redundancy* given by the d_L and θ_M joints. This fact can provoke problems since the system has to be able to *fix* one. This suggests that an entry argument for the positioning IKP could be the current location of the manipulator (i.e. d_L and θ_M values), choosing the closest¹⁵ position in the space of joints (taking into account a free-collision workspace over the table) [41][44].

Let θ_1 - θ_6 be the numerical angle values of the articulations *A1-A6* (see Figure 2.5, right). The IKP consists of finding these values. To practical effects, the operator *uses* these values to *decide* the relocation of the manipulator with regard to the workpiece, by moving the additional joints (θ_M and d_L).

The first three joints have a planar structure that allows obtaining the first three joint values $\{\theta_1, \theta_2, \theta_3\}$. Likewise, the last three joint values $\{\theta_4, \theta_5, \theta_6\}$ orientate the tool and the problem can be solved after determining the first three joint values. With the observations made there is

¹⁴ In the KR15/2 model, the internal configuration of an *status parameter* also controls the preferred posture (a combination of bits which are used to deal with ambiguities in the axis position), see [30].

¹⁵ Nevertheless, the notion of "closeness" could be defined in several different ways. In this particular case, it could be profitable to establish a consideration so that the selection was favoring the movement of the manipulator instead of moving the biggest θ_M and d_L joints, when this option exists. It is due to reasons of precision and economy in the articulate motions.

feasible the resolution of the IKP in order to implement an effective control.

ii) Resolution of the gross positioning

First, it is necessary to obtain the position of the wrist (W , Figure 2.8) when the values that specify the position and orientation of the TCP in the Cartesian working space $\{B\}$ (Figure 2.18) are known with the homogeneous transformation matrix:

$${}^B A_{TCP} = \begin{bmatrix} [{}^B n_{TCP}]_x & [{}^B s_{TCP}]_x & [{}^B a_{TCP}]_x & [{}^B p_{TCP}]_x \\ [{}^B n_{TCP}]_y & [{}^B s_{TCP}]_y & [{}^B a_{TCP}]_y & [{}^B p_{TCP}]_y \\ [{}^B n_{TCP}]_z & [{}^B s_{TCP}]_z & [{}^B a_{TCP}]_z & [{}^B p_{TCP}]_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.45)$$

in which the nomenclature is inherited from Section 2.1.2. In case of mounting a milling tool ($\{T\}$) instead a parallel-jaw gripper, the conventionalism can be applied as depicted in Figure 2.23. In it, the position vector ${}^B p_{TCP}$ points from the origin of $\{B\}$ to the origin of $\{T\}$ (the TCP). Also the n_{TCP} (*normal*), s_{TCP} (*sliding*) and a_{TCP} (*approach*) vectors of $\{T\}$ are depicted in Figure 2.23.

Let ${}^6 A_{TCP}$ be the homogeneous transformation matrix defining the position and orientation of the TCP ($\{T\}$ frame) regarding the robot flange frame ($\{F\}$ frame), see Figure 2.23. This data are directly obtained from the characterization of the tool in memory of the controller. With analogous nomenclature, it is possible to obtain the position and orientation of the robot flange $\{F\}$ with regard to the base frame $\{B\}$ on the table as:

$${}^B A_6 = {}^B A_{TCP} \cdot ({}^6 A_{TCP})^{-1} \quad (2.46)$$

thus [42]

$${}^B p_W = {}^B p_6 - d \cdot z_6 \quad (2.47)$$

is the position of W regarding $\{B\}$. Then, it is necessary to express those coordinates in the robot base frame $\{R\}$ (Figure 2.23) in order to be able to solve the geometric problem. Namely,

$$\begin{bmatrix} {}^R p_W \\ 1 \end{bmatrix} = \begin{bmatrix} d_M \\ [I]_{3 \times 3} d_L \\ [0]_{1 \times 3} h_M \\ 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_M) & -\sin(\theta_M) & 0 & 0 \\ \sin(\theta_M) & \cos(\theta_M) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^B p_W \\ 1 \end{bmatrix} \quad (2.48)$$

In the previous expression, d_M and h_M are constant design values of the workcell whereas d_L and θ_M are the known external joint values. $[I]_{3 \times 3}$ is the identity matrix of size 3.

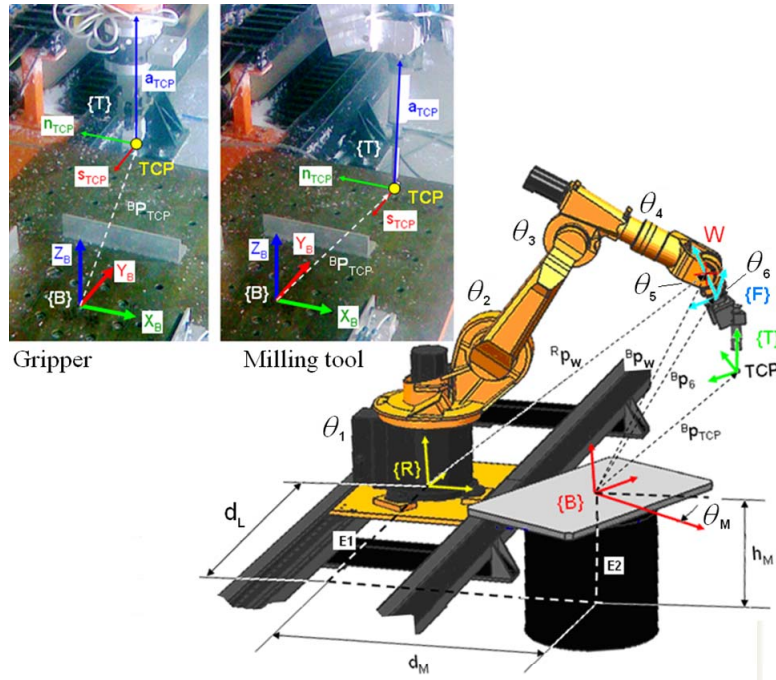


Figure 2.23. Design parameters of the workcell (h_M , d_M) and additional external joint values (θ_M , d_L). Significant position vectors in the workspace are shown.

Given ${}^R p_W = [W_x, W_y, W_z]^T$, the value of the first joint is calculated as:

$$\theta_1 = -\text{atan2}(W_y, W_x) \quad (\text{rad}) \quad (2.49)$$

where the sign of θ_1 is due to the definition of the positive sense of rotation given by the manufacturer.

Preliminary additional calculations are required to obtain the angles θ_2 and θ_3 . According to Figure 2.24, the following parameters can be determined:

$$p = \sqrt{(W_x^2 + W_y^2)} - 300 \tag{2.50}$$

$$h = W_z - 675 \tag{2.51}$$

$$a = \sqrt{(155^2 + 600^2)} \tag{2.52}$$

$$b = 650 \tag{2.53}$$

Thus,

$$\varepsilon = \text{atan2}(h, p) \quad (\text{rad}) \tag{2.54}$$

$$c = \sqrt{h^2 + p^2} \tag{2.55}$$

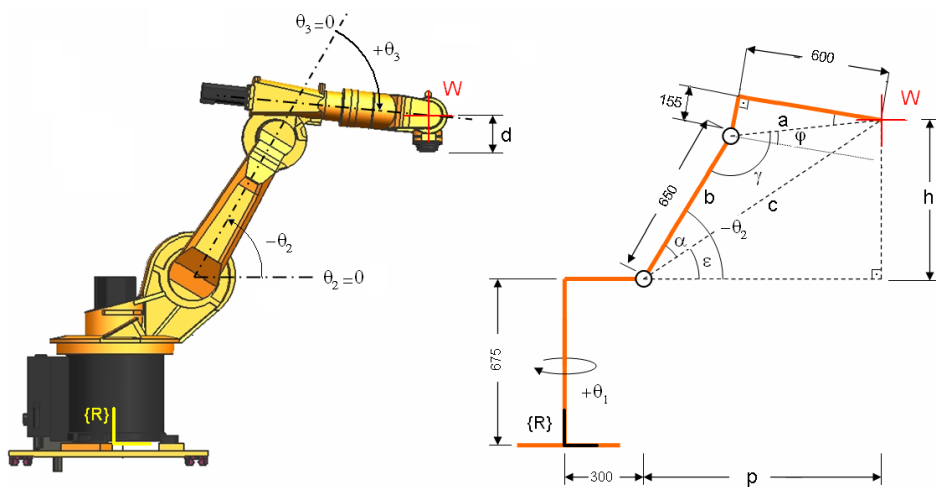


Figure 2.24. The resolution by triangulation in the plane defined by $\{\theta_1, \theta_2, \theta_3\}$. Measurements in millimetres.

Let s be the semiperimeter of the abc triangle, and r the radius of the inscribed circle:

$$s = \frac{a + b + c}{2} \quad (2.56)$$

$$r = \sqrt{\frac{(s-a) \cdot (s-b) \cdot (s-c)}{s}} \quad (2.57)$$

Now α y γ can be identified as:

$$\alpha = 2 \cdot \text{atan}(r/(s-a)) \quad (2.58)$$

$$\gamma = 2 \cdot \text{atan}(r/(s-c)) \quad (2.59)$$

Due to the fact that θ_5 is measured up to the straight line between the joints A3 and A5 and not up to the side a of the abc triangle, it is necessary to consider the correction angle φ :

$$\varphi = \text{atan}(155/600) \quad (2.60)$$

Given α y γ and φ , it is possible to obtain:

$$\theta_2 = -(\alpha + \varepsilon) \quad (rad) \quad (2.61)$$

$$\theta_3 = \pi - \gamma + \varphi \quad (rad) \quad (2.62)$$

where the negative sign of θ_2 is due to the fact that the robot is employed at negative ranges (Figure 2.24).

iii) Resolution of the fine positioning

Let $\{R'_w\}$ be a coordinated system coaxial with $\{R\}$ and linked to the wrist W (Figure 2.25). It is possible to appreciate that, by means of a rotation in z' with value $\rho_1 = -\theta_1$ followed by a rotation in y'' with value $\rho_2 = \pi/2 + \theta_2 + \theta_3$, it is achieved a coordinate system $\{R'''_w\}$ whose axis y''' is coaxial with the axis of the joint θ_5 and with z''' in the direction of the forearm from the joint θ_3 to W. The homogeneous matrix that gives the position and orientation of $\{R'''_w\}$ regarding $\{R\}$ has the following structure:

$${}^R A_{R''W} = \begin{bmatrix} [{}^R R_{R''W}]_{3 \times 3} & [{}^R p_W] \\ [0]_{1 \times 3} & 1 \end{bmatrix} \quad (2.63)$$

where ${}^R A_{R''W}$ will be denoted as ${}^R A_W$ for simplicity. The vector ${}^R p_W$ is known by means of (2.46), and sub-matrix ${}^R R_{R''W}$ is given by

$${}^R R_{R''W} = \begin{bmatrix} \cos(\rho_1) & -\sin(\rho_1) & 0 & 0 \\ \sin(\rho_1) & \cos(\rho_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\rho_2) & \sin(\rho_2) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\rho_2) & 0 & \cos(\rho_2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.64)$$

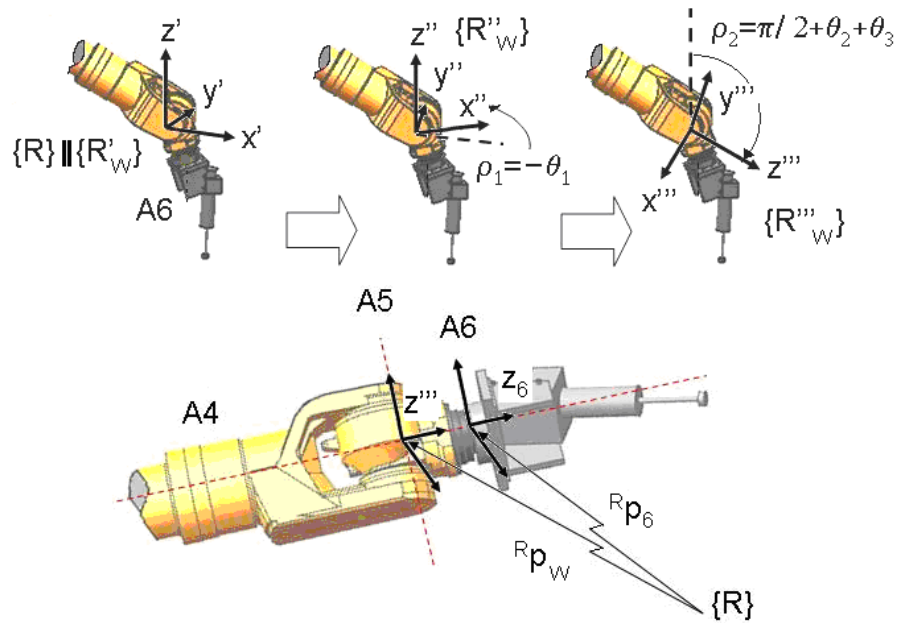


Figure 2.25. Obtaining the coordinate system $\{R'''_w\}$ linked to W

For the following calculations, it becomes necessary to obtain the orientation of the robot flange regarding $\{R'''_w\}$, that is:

$${}^W A_6 = ({}^R A_W)^{-1} \cdot {}^R A_6 \quad (2.65)$$

Thus, the position of the robot flange {F} with regard to {R} is needed (see Figure 2.23). From (2.46), ${}^R A_6$ is obtained as:

$${}^R A_6 = \begin{bmatrix} d_M \\ [I]_{3 \times 3} d_L \\ [0]_{1 \times 3} h_M \\ 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_M) & -\sin(\theta_M) & 0 & 0 \\ \sin(\theta_M) & \cos(\theta_M) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} {}^B A_6 \quad (2.66)$$

According to the way in which the coordinate system $\{R''''_w\}$ has been defined, the axis z'''' makes a distal sense (towards the end of the forearm). The axes of rotation of the last three joints intersect within W and the axes of rotation of the joints θ_4 and θ_6 match with z'''' , whereas the axis of rotation of θ_5 matches with y'''' . Therefore, the angles θ_4 , θ_5 and θ_6 can be deduced from ${}^W A_6$ and the definition matrix of Euler's rotations ZYZ:

$$\theta_4 = -\text{atan2}({}^W A_6(2,3), {}^W A_6(1,3)) \quad (2.67)$$

$$\theta_5 = \text{atan2}(\sqrt{({}^W A_6(3,1))^2 + ({}^W A_6(3,2))^2}, {}^W A_6(3,3)) \quad (2.68)$$

$$\theta_6 = \text{atan2}(-{}^W A_6(3,2), {}^W A_6(3,1)) \quad (2.69)$$

The negative sign of θ_4 is due to the criterion of the mechanical axis of the robot. With these values of the six joints, the inverse kinematics of the robot KUKA KR15/2 is solved for programming control purposes.

2.4.4. Workcell Jacobian

While most trajectory planning methods in Cartesian-coordinate level focus on *position* on the path followed by the operation point (the TCP), the essential inverse kinematic of a six-axis robotic manipulator for milling tasks requires the specification of the *orientation* of the EE as well.

For some simple industrial labours, the position and the orientation tasks are separable; hence, the planning of the two tasks can be done independently. This is the case of usual pick-and-place operations done with common *decoupled* industrial manipulators. This kinematic analysis was introduced at the end of section 2.2.2. iii). More detailed explanation for an isolated KUKA KR15/2 manipulator can be found in [14]. However, *this separation is not possible for most robotic operations, and thus both tasks must be planned concurrently* [1].

Therefore, in the particular case of the workcell studied for milling tasks, (2.17) and (2.18) would become, respectively,

$$v = \dot{\theta}_M e_M \times r_M + v_L e_L + \dot{\theta}_1 e_1 \times r_1 + \dot{\theta}_2 e_2 \times r_2 + \dot{\theta}_3 e_3 \times r_3 \quad (2.70)$$

$$\begin{aligned} \omega &= \omega_M + \omega_1 + \omega_2 + \omega_3 + \omega_4 + \omega_5 + \omega_6 = \\ &= \dot{\theta}_M e_M + \dot{\theta}_1 e_1 + \dot{\theta}_2 e_2 + \dot{\theta}_3 e_3 + \dot{\theta}_4 e_4 + \dot{\theta}_5 e_5 + \dot{\theta}_6 e_6 \end{aligned} \quad (2.71)$$

where it can be appreciated the linear velocity contribution v_L of the track. The 6×8 Jacobian matrix (J) would be completed as done in (2.20), with

$$\begin{aligned} J_{11} &= [e_M \quad 0 \quad e_1 \quad e_2 \quad e_3]; \quad J_{12} = [e_4 \quad e_5 \quad e_6] \\ J_{21} &= [e_M \times r_M \quad e_L \quad e_1 \times r_1 \quad e_2 \times r_2 \quad e_3 \times r_3] \end{aligned} \quad (2.72)$$

$$\dot{\theta} = [\dot{\theta}_M \quad v_L \quad \dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3 \quad \dot{\theta}_4 \quad \dot{\theta}_5 \quad \dot{\theta}_6]^T \quad (2.73)$$

A common misconception in the robotics literature is to confuse J_a , which maps joint rates into the EE location *velocities* (eq. (2.14)), with J_g defined by Whitney and introduced in (2.15), which maps joint rates into the EE *twist* [11]. Thus, the difference between the two Jacobians is essential: J_a is an actual Jacobian matrix, while J_g , properly speaking, is not [1].

From (2.5), a toolpath can be expressed as a function of the form

$$f(q) = s_d \quad (2.74)$$

In order to find J_a in eq. (2.74), by application of the chain rule we get

$$\dot{f} = \frac{\partial f}{\partial q} \dot{q} \equiv J_a \cdot \dot{q} \quad (2.75)$$

From the definition of f , we have that \dot{f} is the time-derivative of the pose array of the EE, i.e., \dot{s}_d . Moreover, it is well known [11] the relation between the two Jacobians, namely

$$J_a = M \cdot J_g \quad (2.76)$$

with M defined as

$$M \equiv \begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix} \quad (2.77)$$

where the sub-matrix Q takes on various forms, depending on the type of rotation representation adopted. Thus, from (2.76) we get

$$J_a \cdot \dot{q} = M \cdot J_g \cdot \dot{q} \rightarrow \dot{s}_d = M \cdot t \quad (2.78)$$

and, therefore, the time-derivative of the toolpath can be expressed as a linear transformation of the twist t of the EE, being it easier to compute with the appropriate J_g , i.e. mapping joint rates into the EE twist (as described at the first part of Section 2.2.2. iii).

2.4.5. Tool-holder characterization

For the resolution of the DKP and the IKP it is necessary to establish the relationship between the *robot flange* and the TCP of the tool, i.e. the ${}^6A_{TCP}$ homogeneous transformation matrix defining the position and orientation of the *TCP* with regard to the robot flange frame (Figure 2.27). On delivery of the robot, the *mechanical* robot flange frame is located as shown in Figure 2.26. Note that the direction assigned to the Z axis is contrary to the rotation sense depicted in Figure 2.5, and therefore it is opposed to the *natural* one of the DH models of Figure 2.20 and Figure 2.21.

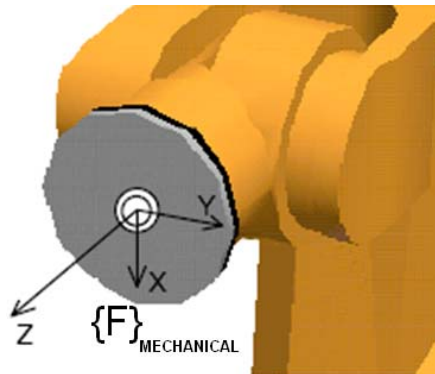


Figure 2.26. Location of the robot flange frame on delivery.

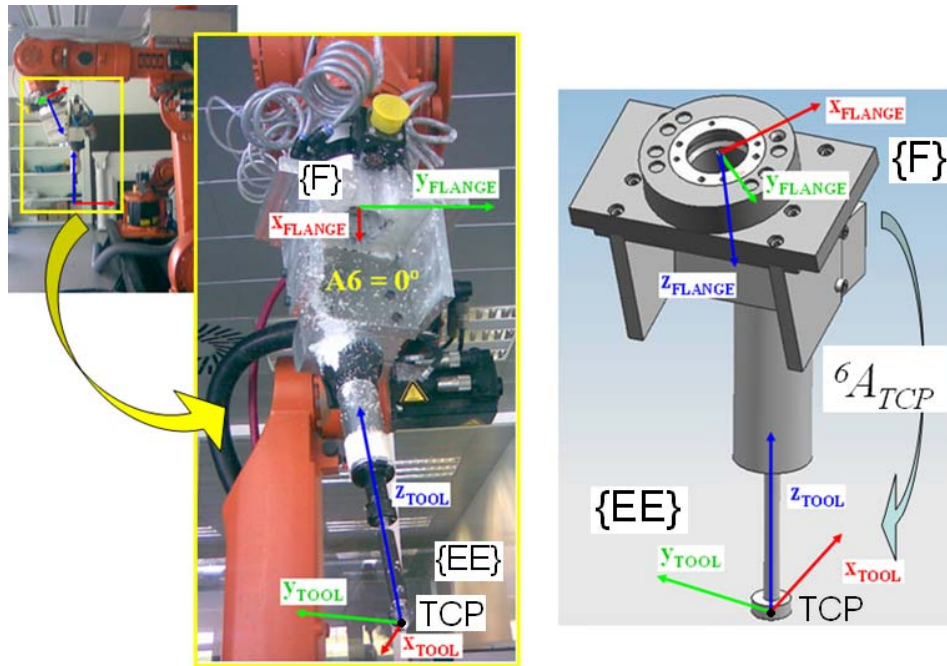


Figure 2.27. Tool holder designed for milling tasks at the IDF.

To practical effects, the KRC2 controller implements a routine to measure the tool by touching a fixed reference point at Ω with the TCP by four different orientations [45]. This measurement method, a calibration of the tool itself, follows the closed-loop chain methods described in Chapter 3. As a result, the controller memorizes the position and orientation of the TCP regarding the robot flange frame with 6 values: the position coordinates and the RPY¹⁶ values (ABC), as shown in Table 2.4.

X -43.30 mm	A -22.67°
Y 17.30 mm	B -19.52°
Z 414.24 mm	C -180.04°

Table 2.4. Position and orientation measured for the tool-holder regarding the robot flange frame.

¹⁶ In the KRC2 controller, the RPY values are defined as three consecutive rotations in Z, Y and X axes, respectively.

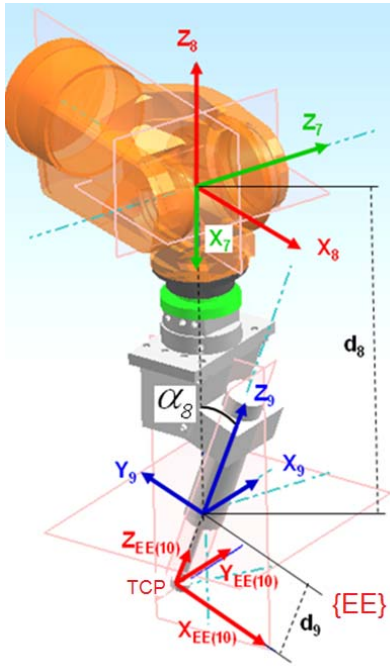
For the calculations considered in previous sections, ${}^6A_{TCP}$ is easily achieved:

$${}^6A_{TCP} = \begin{bmatrix} [{}^6R_{TCP}]_{3 \times 3} & [{}^6p_{TCP}]_x \\ & [{}^6p_{TCP}]_y \\ & [{}^6p_{TCP}]_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.8697 & -0.3856 & 0.3081 & -43.30 \\ -0.3633 & -0.9226 & -0.1294 & 17.30 \\ 0.3341 & 0.0007 & -0.9425 & 414.24 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.79)$$

$$\begin{aligned} {}^6R_{TCP} &= \text{rotz}(A) \cdot \text{roty}(B) \cdot \text{rotx}(C) = \\ \text{with} \quad &= \begin{bmatrix} \cos(A) & -\sin(A) & 0 \\ \sin(A) & \cos(A) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(B) & 0 & \sin(B) \\ 0 & 1 & 0 \\ -\sin(B) & 0 & \cos(B) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(C) & -\sin(C) \\ 0 & \sin(C) & \cos(C) \end{bmatrix} \end{aligned} \quad (2.80)$$

As previously introduced, an additional rotation in the X-axis, $\text{rotx}(\pi)$ would be required to match both *mechanical* and *DH-modelled* {F} frames.

For the scope of this thesis, the full DH model can be directly deduced from Figure 2.21 and Table 2.3, including the particular geometry of the tool, as depicted in Figure 2.28. Measures are directly obtained from the CAD model.



Link	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
1	$\pi/2$	803	θ_M	-305
2	$\pi/2$	0	0	d_L
3	$\pi/2$	300	θ_1	-675
4	0	650	θ_2	0
5	$\pi/2$	155	θ_3	0
6	$\pi/2$	0	θ_4	-600
7	$\pi/2$	0	θ_5	0
8	0.3564	0	θ_6	-443.4
TCP	0	0	0	-119.7

Figure 2.28. and Table 2.5. Robot {EE} frame assignments and parameters for the complete DH model of the IDF workcell, with the tool holder designed for milling purposes.

2.4.6. Characteristic length L of the KUKA KR-15/2

Given the manipulator DH parameters (section 2.4.2.), finding its *characteristic length* L and, hence, its associated minimum condition number is known as the *direct problem* [19], as stated in section 2.3.2. -i).

Prior to the formulation of the optimization problem at hand, it is remarkable that not all DH architecture parameters and not all posture variables influence the condition number adopted [1][19]. In fact, if a subset of the joint variables of a manipulator amount to a rigid-body motion of the overall manipulator, such as in the case of the prismatic joints of a Cartesian manipulator, then these joint variables do not affect the condition number of the manipulator [47]. Also, Khan [19] states that “*the first joint variable of a serial, n-revolute homogeneous manipulator does not influence the condition number (k_F or k_2) of its homogeneous Jacobian; neither do the architecture parameters d_1 and α_n .*”

Therefore, in the case of the first joint variable (θ_M , the rotary table), it is equivalent to a pure translation of the manipulator as if it was a rigid body. The linear track (d_L) also does a pure translation and, thus, it can be left apart when evaluating the conditioning of the manipulator managed. It also matches with the real purpose of the additional external joints, i.e. relocating the manipulator to get a more *convenient* (or *better conditioned*) posture of the $\{\theta_1, \dots, \theta_6\}$ main chain. As a consequence, the straightforward problem of determining the characteristic length of the KR 15/2 will be planned as follows [19]. Taking into account Figure 2.20, let

$$M \equiv \max \{a_M, b_M\} \quad (2.81)$$

with

$$a_M \equiv \max_i \{b_i\}_1^n, \quad b_M \equiv \max_i \{|b_i|\}_2^n \quad (2.82)$$

From table 2.2 and with the value of M , it is possible to redefine a *non-dimensional* DH table of the KR 15/2 as follows

$$\tilde{a}_i \equiv a_i / M, \quad \tilde{b}_i \equiv b_i / M \quad (2.83)$$

for $i = 1, \dots, 6$.

Additionally, let

$$\bar{M} \equiv M/L \quad (2.84)$$

with L being the still unknown *characteristic length*. Hence,

$$\bar{a}_i \equiv a_i/L, \quad \bar{b}_i \equiv b_i/L \quad \rightarrow \quad \bar{a}_i \equiv \tilde{a}_i \cdot \bar{M}, \quad \bar{b}_i \equiv \tilde{b}_i \cdot \bar{M} \quad (2.85)$$

for $i=1, \dots, 6$, which is a set of unknown dimensionless parameters as \bar{M} is still an unknown. Analogously, from the definition of the homogeneous Jacobian matrix (H), eq. (2.38), h_i can be redefined as

$$\bar{\rho}_i \equiv \frac{r_i}{L} = \bar{M} \cdot \tilde{\rho}_i \rightarrow h_i = \begin{bmatrix} e_i \\ e_i \times \bar{M} \cdot \tilde{\rho}_i \end{bmatrix} \quad (2.86)$$

for $i=1, \dots, 6$.

Hence, to find the characteristic length L , all we need is to find the value of \bar{M} that will render the condition number of H a minimum with a suitable set of values for the *last five* joint variables. Let all these design variables be grouped in a design vector \tilde{x} , namely (Figure 2.21)

$$\tilde{x} \equiv \{\bar{M}, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\} \quad (2.87)$$

The value of vector \tilde{x} is found as the solution to the optimization problem:

$$\min_{\tilde{x}} k_F^2(H) \quad , \quad \text{with}^{17} \quad \bar{M} > 0 \quad (2.88)$$

To practical effects, Matlab was used for solving the problem by means of the function `fminsearch`, which uses the Nelder-Mead simplex (direct search) method¹⁸. The algorithm and specific sintaxis can be resumed as follows (Figure 2.29). In each particular configuration of the tool holder, the characteristic length will be obtained by following the same scheme.

For the generic KR 15/2, without any tool attached to the robot flange, the characteristic length obtained was $L=350.6$ mm and the best conditioning achieved was $k_F=1.2477$, which corresponds with the posture depicted in Figure 2.30.

¹⁷ $k_F^2(H)$ is an even function of \bar{M} , $k_F^2(M) = k_F^2(-M)$, and hence if $-\bar{M}$ is a solution to the optimization problem, then so is \bar{M} .

¹⁸ MATLAB Function Reference: `fminsearch`

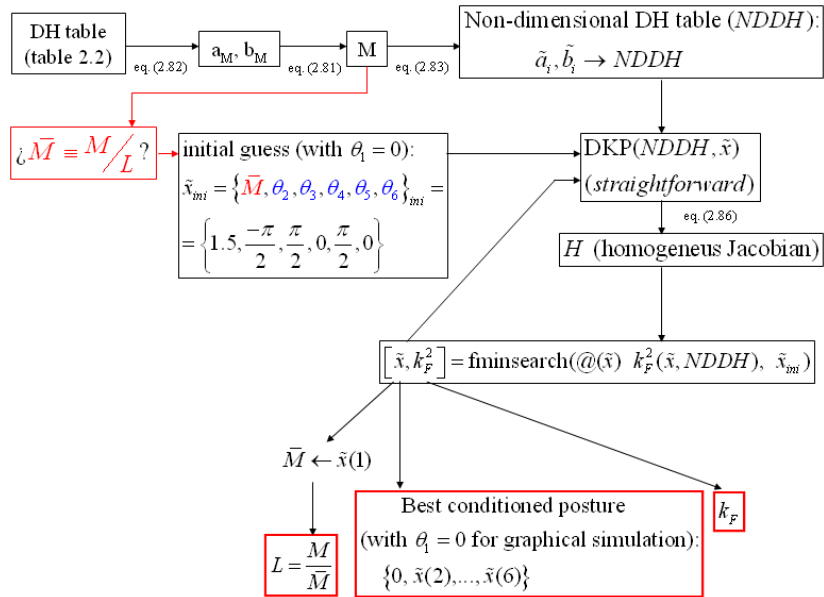


Figure 2.29. Algorithm in Matlab to find the characteristic length (L) of a manipulator. In this particular case, table 2.2 refers to the KR 15/2 without any tool attached, giving a value of $L=350.6$ mm.

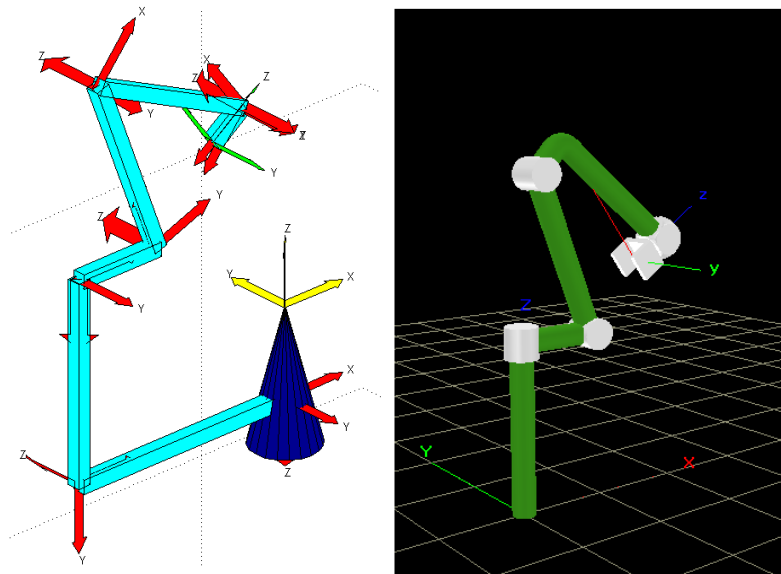


Figure 2.30. Left, best conditioned posture for the 6R KR 15/2 manipulator without any tool attached to the robot flange. Right, same result obtained with the RSV4W [48].

REFERENCES (Ch. 2)

- [1] Angeles, J., Fundamentals of robotic mechanical systems: theory, methods and algorithms, Springer, New York, 521 pages, 2003.
- [2] Golub, G. H. and C. Van Loan, 1989: *Matrix Computations*, Johns Hopkins University Press, 2nd edition, Baltimore, Maryland.
- [3] Arenson, N., Angeles, J. and Slutski, L., Redundancy-resolution algorithms for isotropic robots, *Advances in Robot Kinematics: Analysis and Control*, pp. 425-434, 1998.
- [4] Siciliano, B., Solving manipulator redundancy with the augmented task space method using the constraint Jacobian transpose, *IEEE Intern. Conf. on Robotics and Automation, Tutorial M1*, pp. 5.1-5.8, 1992.
- [5] <http://mathworld.wolfram.com/NullSpace.html>
- [6] Liégeois, A., Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms, *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 245-250, Mar. 1977.
- [7] Yoshikawa, T., Analysis and control of robot manipulators with redundancy, *Robotics Research: The First International Symposium*, pp. 735-747, 1984.
- [8] Pieper, D.L., *The Kinematics of Manipulators under Computer Control*, Ph.D. thesis, Stanford University, 1968.
- [9] Whitney, D.E., Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man-Machine Syst.*, vol. 10, no. 2, pp. 47-53. 1969.
- [10] Ollero; *Robótica: Manipuladores y robots móviles*, Marcombo, 2001. ISBN 8426713130
- [11] Barrientos, Antonio; Peñín, Luis Felipe; Balaguer, Carlos & Aracil, Rafael; *Fundamentos de robótica*, 2ª Ed., Cap. 4. ISBN: 8448156366
- [12] Arenson, N.; *REAL TIME REDUNDANCY-RESOLUTION SCHEMES FOR ROBOTIC MANIPULATORS*, Department of Mechanical Engineering-McGill University, Montréal, 1998
- [13] Tsai, L. W. and Mrogon, A. P., Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. *ASME J. Mechanisms, Transm., and Auto. in Design*, pp. 189-200, 1985.
- [14] Sciavicco, L. and Siciliano, B.; *Modelling and Control of Robot Manipulators*; Springer, London, 2000, pp. 79-87. ISBN 1 85233 221 2 (hbk)
- [15] Whitney, D.E., The mathematics of coordinated control of prosthetic arms and manipulator, *ASME J. Dynamics Systems, Measurement and Control*, Vol. 94, No. 4, pp. 303-309, 1972.
- [16] Osita D.I. Nwokah, Yildirim Hurmuzlu; *The Mechanical systems design handbook; modeling, measurement, and control*; CRC Press, pp. 465-468, 2002. ISBN 0-8493-8596-2
- [17] Luis Gracia, Javier Andres, and Josep Tornero; *Trajectory Tracking with a 6R Serial Industrial Robot with Ordinary and Non-ordinary Singularities*; *International Journal of Control, Automation, and Systems* (2009) 7(1):85-96
- [18] M. D. J. Hayes, M. L. Husty and P. J. Zsombor-Murray; "Singular Configurations of Wrist-Partitioned 6R Serial Robots: a Geometric Perspective for Users", Carleton University (Canada), 2003

- [19] Khan Waseem A.; Angeles J.; The Kinetostatic Optimization of Robotic Manipulators: The Inverse and the Direct Problems; *Journal of Mechanical Design* 2006;128(1):168 - 178.
- [20] Gosselin C.; Angeles J.; A Global Performance Index for the Kinematic Optimization of Robotic Manipulators; *Journal of Mechanical Design* 1991;113(3):220 - 226.
- [21] Angeles J.; The Design of Isotropic Manipulator Architectures in the Presence of Redundancies; *The International Journal of Robotics Research* 1992; 11; 196
- [22] Angeles J., López-Cajún C. S.; Kinematic Isotropy and the Conditioning Index of Serial Robotic Manipulators; *The International Journal of Robotics Research* 1992; 11; 560
- [23] Paul R. P., Stevenson C. N., "Kinematics of Robot Wrists", *The International Journal of Robotics Research*, Vol. 2, pp. 31-38, 1983.
- [24] Tsai Y. C., Soni A. H., "Accessible Region Synthesis of Robot Arms", *ASME, Journal of Mechanical Design*, Vol. 103, pp. 803-811, 1981.
- [25] Yang, D.C., and Lai, Z.C.; On the conditioning of robotic manipulators-service angle. *ASME J. Mechanisms Transmissions and Autom. Des.* 107:262-270 (1985)
- [26] A. Kumar and K.J. Waldrom, "The Workspace of a Mechanical Manipulator", *ASME J. of Mechanical Design* 103, 665-672 (1981).
- [27] T. Yoshikawa: "Manipulability of Robotic Mechanisms", *Int. J. of Robotics Research*, Vol.4, No.2, pp.3-9, 1985.
- [28] Yoshikawa T. Translational and rotational manipulability of robotic manipulator. In: *American control conference*, S. Diego, CA; 1990.
- [29] Ranjbaran, E, Angeles, J., Gonzalez-Placios, M. A. and Patel, R. V., "The mechanical design of a seven-axes manipulator with kinematic isotropy", *J. of Intelligent and Robotic Systems*, Vol. 14, No. 1, pp. 21-41 (1995)
- [30] KUKA System Software (KSS), *Expert Programming (KRC2 / KRC3)*, Release 5.2., KUKA Corp., 2005.
- [31] Angeles, J. and Rojas, A., 1987, "Manipulator kinematic inversion via condition-number minimization and continuation," *The International J. Robotics & Automation*, Vol. 2, No. 2, pp. 61-69.
- [32] Chiaverini S., Oriolo G., Walker I. D.; "Kinematically Redundant Manipulators". *Springer Handbook of Robotics*, pp. 245-268, (2008).
- [33] KUKA Robot Group; "*KR C2 edition05 - Specification*"; Issued: 20.07.2007 Version: 2.1
- [34] Hartenberg R. S. and Denavit J., "A kinematic notation for lower pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 77, pp. 215-221, June 1955.
- [35] Craig J.J., "Introduction to Robotics". AddisonWesley, second ed., 1989.
- [36] Corke P.I., *A Robotics Toolbox for MATLAB*; IEEE Robotics and Automation Magazine, issue 1, vol. 3, pp. 24-32, march 1996
- [37] Maza, J.I. and Ollero A., *Herramienta MATLAB-Simulink para la simulación y el control de robots manipuladores y móviles*, Actas de las XXI Jornadas de Automática, Sevilla, 2000
- [38] C.S.G. Lee, M. ZIEGLER; *Geometric Approach in Solving Inverse Kinematics of Puma Robots*; Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109; 0018-9251/84/1100-0695

- [39] H. Mayer, I. Nagy, A. Knoll; Inverse Kinematics of a Manipulator for Minimally Invasive Surgery; Institut für Informatik-Technischen Universität München; TUM-INFO-01-I0404-0/1.-FI, January 04
- [40] Tsai, L. W. and Mrogan, A. P., Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. ASME J. Mechanisms, Transm., and Auto. in Design, pp. 189-200, 1985.
- [41] J. J. Craig, Introduction to Robotics. Addison Wesley, second ed., 1989.
- [42] Pieper, D.L., The Kinematics of Manipulators under Computer Control, Ph.D. thesis, Stanford University, 1968.
- [43] R. P. Paul, Robot Manipulators: Mathematics, Programming, and Control. Cambridge, Massachusetts: MIT Press, 1981.
- [44] A. Ollero; "Robótica: Manipuladores y robots móviles", Marcombo, 2001. ISBN 8426713130
- [45] KUKA Corp., 2005. *KUKA System Software (KSS), Puesta en servicio (KRC2 / KRC3), Release 5.2.*
- [46] KUKA Corp., 2005. *KUKA System Software (KSS): Expert Programming (KRC2 / KRC3), Release 5.2.*
- [47] Angeles A. and Rojas A.; Manipulator inverse kinematics via condition-number minimization and continuation; Int. J. of Robotics and Automation, Vol.2, No. 2, pp. 61-69; 1987.
- [48] Khan W., Zhuang H., Angeles J.; "Robot Visualization System for Windows (RVS4W) - User's Manual"; Centre for Intelligent Machines (CIM), Department of Mechanical Engineering, McGill University, Montreal, Quebec, Canada, 2007.

CHAPTER 3

WORKCELL CALIBRATION

*“Have no fear of
perfection, you’ll never
reach it.” –
Salvador Dali*

CHAPTER 3. WORKCELL CALIBRATION

3.1. CONCEPTS ON ACCURACY CRITERIA AND ERROR SOURCES

The international standard ISO 9283 [1] sets different performance criteria for *industrial manipulators* and suggests test procedures in order to obtain appropriate parameter values. The most important criteria, and also the most commonly used, are *accuracy of pose* (AP) and *repeatability of pose* (RP). It is well known that industrial robots have high RP but not AP [2] (Figure 3.1).

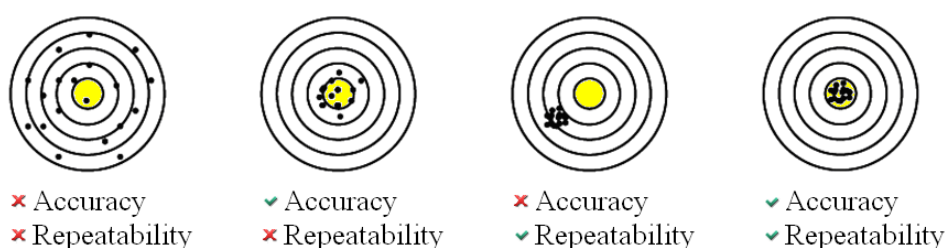


Figure 3.1. Repeatability and accuracy issues for a commanded target.

Repeatability is particularly important when the robot is moved towards the command positions manually (*Teach-In*, e.g. *pick and place* operations). Most of the robotic applications that capitalize on repeatability have already been done. However, if the robot program is generated by a 3D simulation absolute accuracy is particularly relevant (i.e., *off-line* programming of advanced precision applications, such as milling or even robotic surgery.). Both are generally influenced in a negative way by kinematic factors. Here especially the joint offsets and deviations in lengths and angles between the individual robot links take effect.

It has been shown that as much as 95% of robot positioning inaccuracy arises from the inaccuracy in its kinematic model description, i.e., the parameters used to compute the position and orientation [3]. Consequently, a simple, fast and accurate robot geometric model adjusted through a *calibration process* is needed. The position and orientation of the robot is usually determined using forward kinematics with Denavit–Hartenberg (DH) parameters for each link of the robot.

Thus, robot accuracy ultimately depends on the accuracy of these DH parameters.

Robot kinematic calibration is defined as a “process of improving robot’s EE positioning accuracy through modification of its kinematic control model without changing its hardware configurations” [4]. Basically it consists in identifying the differences between geometrical parameters of models given by manufacturers and those of the real robot. These differences come from imprecise knowledge of robot geometry: link length, angle between successive axes, joint off-sets, gear ratio. Some variation comes from the manufacturing process, primarily due to machining inaccuracy. Other variation comes from the assembly process, where the precise position and orientation of links and joints is not perfectly repeatable. Most manufacturers of robots do not focus on accuracy because if accuracy is achieved by higher tolerance in machining, the cost of robot increases spectacularly, affecting the sales negatively.

As a consequence, a calibration approach to identify the DH parameter values is needed to advance the state of the art in robotics. After a calibration procedure, the robot controller can be updated with the correct robot-specific DH parameter values instead of the standard design values. This calibrated robot has a higher absolute positioning accuracy than an uncalibrated one, i.e., the real position of the robot’s EE corresponds better to the position calculated from the mathematical model of the robot.

3.1.1. State of art in robotic calibration

There has been considerable research in the field of robotic calibration, as stated in [5]. Existing techniques can be classified into *open-loop* (or *pose-measuring*) and *closed-loop* (or *pose-matching*) approaches [6].

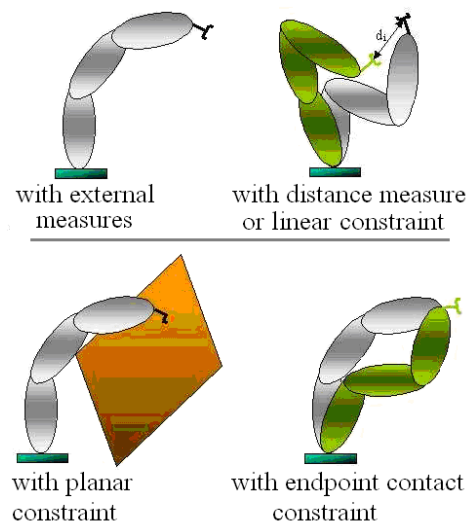


Figure 3.2. Several *open (up)* and *closed (down) loop* methods for robot calibration.

On one hand, conventional *open-loop methods* involve measuring the EE pose, which traditionally requires expensive and complicated pose measuring devices (such as theodolites [7], inclinometers, coordinate measuring machines [8][9], sonic and visual sensors [10][11], and laser tracking system [12][13]). Therefore, the resolution of measurements near the EE is limited by the equipment used. It has been reported that partial pose information is sufficient for complete parameter identification [3]. Generally, these calibration methods involve the following procedures [4] (see Figure 3.3):

- a) Choose a proper kinematic model to describe the relationship between robot joint space \mathfrak{Z} and its operational coordinates at Ω ;
- b) Take experimental measurements of robot EE locations using external measuring devices (the requirements of measurement phase are particularly demanding);
- c) Identify the parameter errors based on the differences between the measured locations and those predicted by the kinematic model.
- d) Implement the identified model in the robot controller (i.e., compensation).

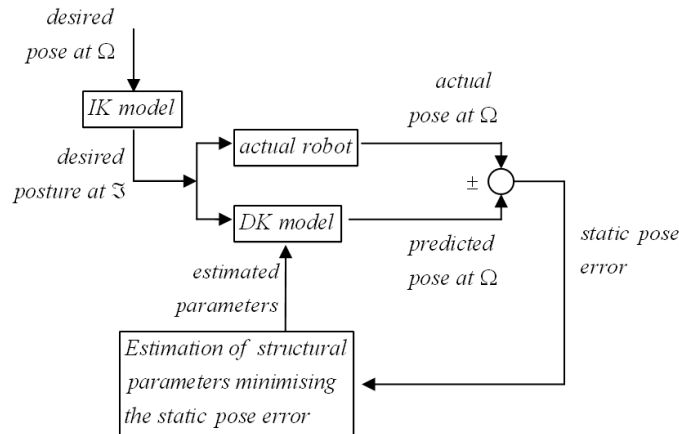


Figure 3.3. Generic procedure in a robot calibration using an *open-loop* method [6]

On the other hand, *closed-loop methods* are defined as the automated process of determining a robot's model by using only its internal sensors, and thereby, can be named *self-calibrating* or *autonomous* methods [3][14][15]. It has been observed that autonomous calibrations are possible for robot manipulators with either some a priori knowledge of the task constraint. These methods impose some constraints or some sort of motion on the EE, and the joint readings alone are used to calibrate the robot using kinematic closed-loop equations. A standard procedure is depicted in Figure 3.4.

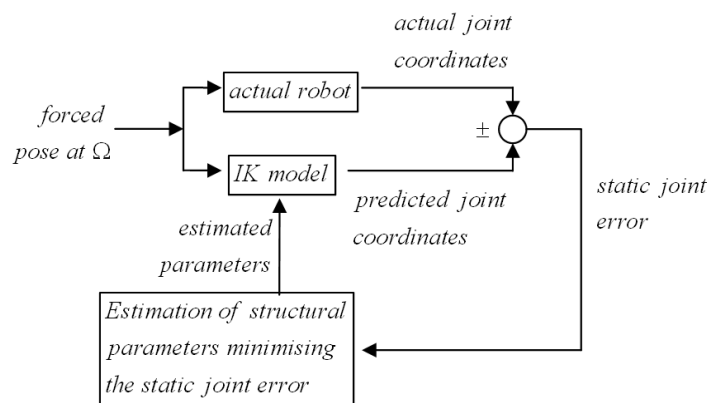


Figure 3.4. Generic procedure in a robot calibration using a *closed-loop* method [6]

Some researchers in the past have used linear constraints allowing the EE to slide along a straight line, e.g., Newman et al. [14] used a laser line tracking in

the robot workspace. Bennett and Hollerbach [16] and Meggiolaro et al. [17] proposed a closed-loop calibration method in which the robot endpoint is fixed at a single point contact constraint, equivalent to a ball joint. In that case, the robot moves to different configurations that satisfy the contact constraint. However, it is difficult to move a physically closed kinematic chain from one posture to another while maintaining the physical constraints. Hence, it is difficult to collect accurate joint readings. A profuse description of the algorithms associated to the methods constraining the EE is carried out by Khalil et al. [18][19][20]. Gatla et al. [5] proposed a *Virtual Closed Kinematic Chain* method, in which the approach does not require any physical constraint. In this case, a laser tool is attached to the EE. This laser tool aims at an arbitrary but fixed point on a distant plane; thus, creating a virtual closed kinematic chain. The calibration procedure requires many different robot joint configurations and the process can be time-consuming. The authors propose a feedback system in which a camera detects the laser spot on the plane and it is used as feedback to adjust the robot joint angles so as to move the laser spot to the desired fixed point.

Ikits and Hollerbach [21] extended their *Implicit Loop Method* to the use of a *planar constraint*, where the robot endpoint is constrained to lie on a plane. Zhuang et al. [22] also imposed plane constraints on the EE positions. These authors showed that a single-plane constraint is normally insufficient for calibrating a robot. It was also shown that by using a three-plane constraint, the constrained system is equivalent to an unconstrained point-measurement system under certain conditions. The significance of this result is that the three-plane constraint setup can be used to successfully calibrate a robot. A profuse description of the planar methods is carried out by Khalil and his co-workers [19][20][23]. In the *planar constraint* methods, the use of a contact probe is problematic because it is difficult to be certain that the tip-point of the probe is exactly on the surface, neither above it nor indenting it.

Based on the *planar constraint procedure* and the *open-loop methods*, a new robot kinematic calibration scheme is presented at Section 3.3. It can be implemented autonomously and is suitable for *on-site calibration in an industrial environment at regular intervals*, in contrast with other open-loop methods requiring extensive human intervention and expensive or demanding devices such as those previously mentioned. By holding a laser displacement sensor, the robot sweeps three orthogonal constraint planes in its workspace while measuring the distance, which is supposed to be constant. Only the distance readings are recorded. A *non-linear least-squares* (NLSQ) identification model has been derived from the consistency conditions of the planes, and is presented.

3.2. CALIBRATION. MATHEMATICAL BACKGROUND

The robot errors gathered by pose measurements can be minimized by numerical optimization. For kinematic calibration, a complete kinematical model of the geometric structure must be developed, whose parameters then can be adjusted by mathematical optimization.

In practice, for the general problem of calibrating a mechanical system, it can be shown that the main objective of the procedure consists on determining a best approximation of a calibration matrix, C , by taking many data samples of the *actual* variables and the *real sensed* variables:

$$[Actual] = [C][Sensed] \quad (3.1)$$

Given a large number of sensor readings and the corresponding actual inputs, *least-squares* descent methods fitting these data points have demonstrate its convenience for solving the kinematical optimization problems [19][20][24]. The method of least-squares is used to approximately solve such overdetermined systems, i.e. systems of equations in which there are more equations than unknowns. This is done by creating a single matrix equation from linear approximations of the relationships between sensor responses and actual inputs.

This procedure supplies corrected kinematical parameters for the measured machine, which then for example can be used to update the system variables in the controller in order to adapt the used robot model to the real kinematics.

3.2.1. Problem statement

The objective consists of adjusting the parameters of a *model function* so as to best fit a *data set*. A simple data set consists of np points (data pairs) (q^i, p_s^i) , $i = 1, \dots, np$; where q^i is an independent variable and p_s^i is a dependent variable whose value is found by observation. This *model function* has the form

$$p_M^i = f(q^i; \beta_1, \dots, \beta_{mp}) = f(q^i, \beta) \quad (3.2)$$

where the mp adjustable parameters are held in the vector β . It is desirable to find those parameter values for which the model best fits the data. In summary, the common behaviour of the system to adjust can be described with the vector model function as well as the input and output vectors, as described:

$$\begin{aligned}
q &= (q^1, \dots, q^{np})^T, \quad q^i \in R^k; \quad q \in R^{np \times k} \\
\beta &= (\beta_1, \dots, \beta_{mp})^T, \quad \beta \in R^{mp} \\
p_M &= (p_M^1, \dots, p_M^{np})^T, \quad p_M^i = (p_{M1}^i, \dots, p_{Ml}^i) = f(q^i, \beta) \in R^l, \quad p_M \in R^{np \times l} \\
p_S &= (p_S^1, \dots, p_S^{np})^T, \quad p_S^i \in R^l, \quad p_S \in R^{np \times l}
\end{aligned} \tag{3.3}$$

The variables k , l , mp , np describe the dimensions of the single vector spaces. For a common manipulator we have:

$$\begin{aligned}
i &= 1, \dots, np; \text{ number of observations,} \\
t &= 1, \dots, k; \text{ number of joints associated to each } i\text{-observation (at } \mathfrak{T}) \\
&\quad \text{(e.g., } (q_1^i, q_2^i) \equiv (\theta_1^i, \theta_2^i) \text{ for a 2-DOF manipulator),} \\
h &= 1, \dots, l; \text{ (} l \leq 6 \text{) DOF's of the EE observed at } \Omega \text{ (e.g. } xyzABC), \\
j &= 1, \dots, mp; \text{ number of model parameters to adjust.}
\end{aligned}$$

A residual r is defined as the difference between the observed values of the dependent variable and the predicted values from the estimated model,

$$r^i = p_S^i - f(q^i, \beta); \quad r^i = (r_1^i, \dots, r_l^i)^T \in R^l \tag{3.4}$$

Minimization of the residual error r^i for the purpose of identification of the optimal parameter vector β follows from the difference between both output vectors using the Euclidean norm. The least-squares method defines best as when the sum, S^i , of squared residuals is a minimum.

$$S^i = \|p_S^i - p_M^i\|^2 = \|r^i\|^2 = \sum_{h=1}^l (r_h^i)^2; \quad S^i \in R \tag{3.5}$$

Such in the case of robot calibration, a data point may consist of more than one independent variable. In the most general case there may be one or more independent variables and one or more dependent variables at each data point.

3.2.2. Solving the least-squares problem

Least-squares problems fall into two categories, *linear* and *non-linear*. The linear least-squares problem has a closed form solution, but the non-linear problem does not and is usually solved by iterative refinement; at each iteration the system is approximated by a linear one, so the core calculation is similar in both cases.

The minimum of the sum of squares is found by setting the gradient to zero. Since the model contains mp parameters, there are mp gradient equations for each of the np data pairs (q^i, p_s^i) .

$$\frac{\partial S^i}{\partial \beta_j} = 2 \sum_i r^i \frac{\partial r^i}{\partial \beta_j} \quad (3.6)$$

and substituting (3.4) the gradient equations become

$$\frac{\partial S^i}{\partial \beta_j} = -2 \sum_i \frac{\partial f(q^i, \beta)}{\partial \beta_j} r_i = 0 \quad (3.7)$$

The gradient equations apply to all least-squares problems. Each particular problem requires particular expressions for the model and its partial derivatives.

i) Linear least-squares

A regression model is a linear one when the model function comprises a linear combination of the parameters, i.e.

$$p_M^i = f(q^i, \beta) = \sum_{j=1}^{mp} \beta_j \phi_j(q^i) \quad (3.8)$$

where the coefficients, ϕ_j , are functions of q^i .

Letting

$$J_{f_{hj}}^i = \frac{\partial p_{Mh}^i}{\partial \beta_j} = \frac{\partial f_h(q^i, \beta)}{\partial \beta_j} = \phi_{hj}(q^i) \in R^{k \times mp} \quad (3.9)$$

be the Jacobian matrix of the model function (J_f), it can be shown that, in this case, the least-square estimated β is given by

$$\beta = (J_f^T J_f)^{-1} J_f^T p_M \quad (3.10)$$

ii) Non-Linear Least-Squares (NLSQ)

In a non-linear system, the derivatives $\partial r^i / \partial \beta_j$ are functions of both the independent variable q^i and the parameters β , so these gradient equations do not have a closed solution. Thus, there is no closed-form solution to a NLSQ problem. Instead, initial values must be chosen for the mp parameters and numerical algorithms are used to find the value of the parameters β which minimize the objective. Then, the parameters are refined iteratively, that is, the values are obtained by successive approximation:

$$\beta \approx \beta^{(s+1)} = \beta^{(s)} + \Delta\beta \quad (3.11)$$

where s is an iteration number and the vector of increments, $\Delta\beta$, is known as the *shift vector*.

In commonly used algorithms [19][20][24], at each iteration the model is linearized by approximation to a first-order Taylor series expansion about $\beta^{(s)}$.

$$f(q^i, \beta) \approx f(q^i, \beta^{(s)}) + \sum_j \frac{\partial f(q^i, \beta^{(s)})}{\partial \beta_j} (\beta_j - \beta_j^{(s)}) \quad (3.12)$$

Substituting at (3.4), the residuals are given by

$$r^i = p_s^i - \left(f(q^i, \beta^{(s)}) + \sum_j \frac{\partial f(q^i, \beta^{(s)})}{\partial \beta_j} (\beta_j - \beta_j^{(s)}) \right) \quad (3.13)$$

In terms of this linearized model, for the l components of r , we define

$$J_{r_{hj}}^i \equiv \frac{\partial r_h^i}{\partial \beta_j} = -\frac{\partial f_h(q^i, \beta^{(s)})}{\partial \beta_j} = -J_{f_{hj}}^i; \quad J_r^i, J_f^i \in R^{l \times mp} \quad (3.14)$$

For each observed i -point, the Jacobian of the model function, J_f , is a function of constants (the independent variables and the estimated parameters) so it changes from one iteration to the next. Thus,

$$r_h^i = \Delta p_h^i + \sum_{j=1}^{mp} J_{r_{hj}}^i \cdot \Delta \beta_j = \Delta p_h^i - \sum_{j=1}^{mp} J_{f_{hj}}^i \cdot \Delta \beta_j; \quad h = 1, \dots, l \quad (3.15)$$

with

$$\Delta p^i = p_s^i - f(q^i, \beta^{(s)}) \in R^l \quad (3.16)$$

$$\Delta \beta = (\beta - \beta^{(s)}) \in R^{mp} \quad (3.17)$$

Substituting expressions (3.14) and (3.15) into the gradient equations (3.7) and setting the gradient to zero to get the minimum of the sum S of squared residuals (3.4), they become:

$$\left\{ \begin{array}{l} \frac{\partial S^i}{\partial \beta_j} = 2 \sum_h r_h^i \frac{\partial r_h^i}{\partial \beta_j} = -2 \sum_{h=1}^l J_{f_{hj}}^i \left(\Delta p_h^i - \sum_{t=1}^{mp} J_{f_{ht}}^i \cdot \Delta \beta_j \right) \\ \partial S^i / \partial \beta_j = 0 \end{array} \right\} \quad (3.18)$$

which, on rearrangement, become mp simultaneous linear equations, named as the *normal equations*.

$$\sum_{h=1}^l \sum_{t=1}^m J_{f_{hj}}^i J_{f_{ht}}^i \Delta \beta_t = \sum_{h=1}^l J_{f_{hj}}^i \Delta p_h^i \quad (3.19)$$

The *normal equations* are written in matrix notation as

$$(J_f^T J_f) \Delta \beta = J_f^T \Delta p \quad (3.20)$$

The superscript T denotes the matrix transpose. These equations form the basis for the *Gauss-Newton algorithm* for a NLSQ problem.

3.2.3. Gauss-Newton algorithm and its application to calibration algorithms

The *Gauss-Newton algorithm* is a method used to solve NLSQ problems. It can be seen as a modification of Newton's method for finding a minimum of a function. Unlike Newton's method, the Gauss-Newton algorithm can only be used to minimize a sum of squared function values, but it has the advantage that second derivatives, which can be challenging to compute, are not required.

i) Description of the iterative method

For calibration purposes, the above described algorithm can be extended when np functions r^1, \dots, r^{np} of mp variables $\beta = (\beta_1, \dots, \beta_{mp})$ are given, corresponding to np observations done (with $np \geq mp$). The Gauss–Newton algorithm finds iteratively the minimum of the sum of squares

$$S = \sum_{i=1}^{np} S^i(\beta) = \sum_{i=1}^{np} \sum_{h=1}^l r_h^i(\beta)^2; \quad S \in R; \quad \min_{\forall i} \{S\}; \quad (3.21)$$

Starting with an initial guess $\beta^{(0)}$ for the minimum, the method proceeds by the iterations

$$\beta^{(s+1)} = \beta^{(s)} + \Delta\beta \quad \rightarrow \quad \Delta\beta = \beta^{(s+1)} - \beta^{(s)} \quad (3.22)$$

where the increment $\Delta\beta$ is the solution to the normal equations:

$$(W^T W)\Delta\beta = -W^T r \quad (3.23)$$

The goal is to find the parameters β such that a given model function $p_M = f(q, \beta)$ fits best some data points (q, p_S) , eq. (3.1). In robot calibration, r is the vector of functions r^i , e.g. the residuals of the position and/or orientation. For each estimation of $\beta^{(s)}$

$$r(\beta^{(s)}) \rightarrow \Delta p^{(s)} = (\Delta p^{1(s)}, \dots, \Delta p^{n(s)})^T = p_S - f(q, \beta^{(s)}) \quad (3.24)$$

To identify $\Delta\beta$, this equation can be performed for a sufficient number of np configurations, q .

$$q = (q^1, \dots, q^n)^T \quad (3.25)$$

The resulting linear system of equations will be represented by:

$$\Delta p = W(q, \beta)\Delta\beta \quad (3.26)$$

W is the *Observation Matrix* [19] of dimension $(np \cdot l \times mp)$ with $np \gg mp$.

$$W = \begin{bmatrix} J_f(q^1, \beta^{(s)}) \\ \dots \\ J_f(q^n, \beta^{(s)}) \end{bmatrix} \quad (3.27)$$

being it a composition of the J_f Jacobian matrices of the model function, $p_M = f(q, \beta^{(s)})$, for each observation, evaluated at $\beta^{(s)}$.

Then, the increment $\Delta\beta$ can be solved to get the least-squares errors solution as

$$\Delta\beta = W^\dagger \Delta p^{(s)} \quad (3.28)$$

where W^\dagger is the left pseudo-inverse¹ of W .

$$W^\dagger \equiv (W^T W)^{-1} W^T \quad (3.29)$$

The geometric parameters β can be updated, eq. (3.22), after (3.28). By iteratively applying this procedure until the elements of $\Delta\beta$ become smaller than some prescribed limit. Therefore, a *best-fit* solution is obtained. The common sense criterion for convergence is somewhat arbitrary, as for example

$$\left| \frac{\Delta\beta_j^{(s)}}{\sum_s \Delta\beta_j} \right| < 0.001, \quad j = 1, \dots, mp \quad (3.30)$$

which is equivalent to specify that each parameter should be refined to 0.1% precision. This is reasonable when it is less than the largest relative standard deviation on the parameters.

The procedure may not converge very well for some functions and it is often greatly improved by picking the initial value, $\beta^{(0)}$, close to the best-fit value. Equation (3.29) assumes that the inverse of the matrix $W^T W$ exists, as it can be considered for the postures adopted in a calibration procedure within the workspace range (for the scope of this thesis, over the

¹ In fact [33], it verifies that $W^\dagger \cdot W = I$

table as depicted in following sections). A discussion of the singularity of this matrix is given in [25].

ii) Outline of the NLSQ Model-based calibration methods. 2D planar calibration example

The aim of the calibration of the Denavit-Hartenberg (DH) geometric parameters by iterative methods is to minimize the difference between the measured EE location and the calculated location. Rearranging (3.20), a linear differential model defining the deviation of the EE location due to the differential error in the geometric parameters can be expressed as:

$$\Delta p = J_f \Delta \beta \quad (3.31)$$

where:

- $\Delta \beta$ defines the $(mp \times 1)$ vector of the errors of the geometric DH parameters to be adjusted.
- Δp represents the $(l \times 1)$ vector of the position and orientation error (difference between measured and calculated) with $l \leq 6$ DOF's in the Cartesian Operational Space Ω .
- $J_f(q, \beta)$ is the $(l \times mp)$ Jacobian matrix of the homogeneous position matrix of the EE with respect to the DH geometric parameters.

Eq. (3.31) gives l linear equations in the unknown vector $\Delta \beta$.

In classical calibration methods, we need an accurate external sensor to measure the real EE position or location. If this sensor gives only the position coordinates of the EE, then only the first three equations of relation (3.31) will be used. In this case $3n \gg mp$ observations are taken as general rule. Analogously, in a planar motion $2n \gg mp$ observations are taken. The identifiable parameters must be carried out on the corresponding observation matrix.

The exposed calculations can be tested in the calibration of a 2D planar manipulator. This mechanism can represent the two links defining the gross positioning in the KUKA KR 15/2 manipulator (Figure 3.5).

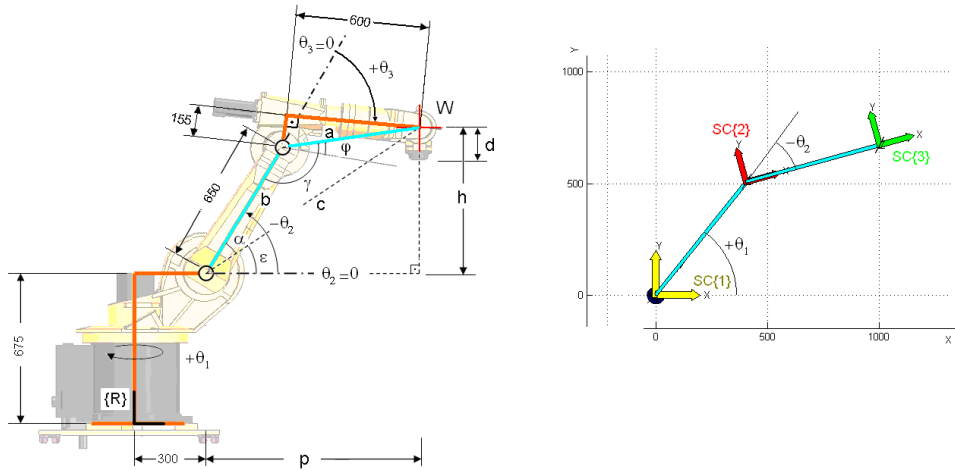


Figure 3.5. 2D planar manipulator of two links simulating the gross positioning in the KUKA KR 15/2 manipulator.

The forward kinematics of the robot is achieved by the DH method (see Chapter 2),

$$p = f(q, \beta) \quad (3.32)$$

where $q = [\theta_1 \ \theta_2 \ \theta_3]^T$ are the joint values and β the vector of the DH parameters,

$$\beta = [\alpha, a, \theta, d] \quad (3.33)$$

which are shown in Table 3.1. These values will be referred as $\beta^{(0)}$ in the iterative process of calibration.

i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
1	0	$a_1 = 650$	θ_1	0
2	0	$a_2 = \sqrt{600^2 + 155^2}$	θ_2	0

Table 3.1. DH parameters for the mechanical system simulating the manipulator gross positioning.

In this case, only the position coordinates in the plane of the motion, $p = [x \ y]^T$, are observed (Figure 3.5). Consequently, only the first

two equations of (3.31) will be used. Both coordinates are resumed from the overall 4x4 DH homogeneous transformation matrix:

$$T = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1 c_1 + a_2 c_{12} \\ s_{12} & c_{12} & 0 & a_1 s_1 + a_2 s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.34)$$

with

$$\begin{aligned} c_1 &= \cos(\theta_1) & s_2 &= \sin(\theta_2) \\ c_2 &= \cos(\theta_2) & c_{12} &= c_1 c_2 - s_1 s_2 \\ s_1 &= \sin(\theta_1) & s_{12} &= c_1 s_2 + s_1 c_2 \end{aligned} \quad (3.35)$$

Then, in this case the theoretical positioning values, p_o , are $T(1,4)$ and $T(2,4)$: It is easy to appreciate that they are non dependent on θ_3 :

$$p_o = \begin{bmatrix} x_o \\ y_o \end{bmatrix} = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \end{bmatrix} \quad (3.36)$$

In each link of the model, it can be considered an existing error. This error is propagated through the forward kinematics giving as a result a positioning error, being it the residual to minimize. Care must be taken to all of the DH parameters: none is assumed to be 0 and left out of the kinematic model or, on the contrary, the error in that parameter is not revealed itself in the calibration process. The identifiable parameters must be carried out on the corresponding observation matrix W , eq. (3.27).

For the purpose of this example, let's consider both errors of 0.2 mm at link a_1 and 1° at θ_1 :

$$\Delta\beta_E = \begin{bmatrix} a_{1E} \\ \theta_{1E} \end{bmatrix} = \begin{bmatrix} 0.2 \text{ mm} \\ 0.0175 \text{ rad } (\approx 1^\circ) \end{bmatrix} \quad (3.37)$$

With this supposition, there are two parameters to adjust:

$$\Delta\beta \equiv \begin{bmatrix} \Delta a_1 \\ \Delta\theta_1 \end{bmatrix} \quad (mp=2) \quad (3.38)$$

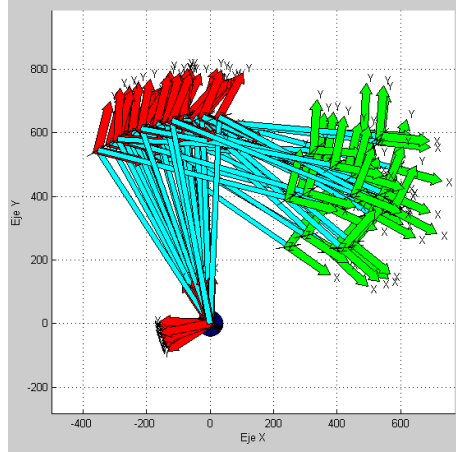


Figure 3.6. Thirty observed points in the workspace

Having in this case an $mp=2$, $np=30$ well-known locations in the workspace, $p_M = (p_M^1, \dots, p_M^{30})^T$, constitute an enough number of configurations to carry out the calibration procedure. If those locations were commanded to the robot controller, the requested configurations $q = (q^1, \dots, q^{30})^T$ would be internally calculated by means of the *IK* theoretical model programmed. In order to emulate the possible configurations of the real KUKA when working on the rotary table, only the *elbow-up* solution is considered (Figure 3.6, see Chapter 2).

$$\begin{aligned} c_2^i &= ((x_s^i)^2 + (y_s^i)^2 - a_1^2 - a_2^2) / (2 \cdot a_1 \cdot a_2) \\ s_2^i &= -\sqrt{1 - (c_2^i)^2} \\ q &= \begin{bmatrix} \theta_1^i \\ \theta_2^i \end{bmatrix} = \begin{bmatrix} \text{arctg}2(y_s^i, x_s^i) - \text{arctg}2(a_2 \cdot s_2^i, a_1 + a_2 \cdot c_2^i) \\ \text{arctg}2(s_2^i, c_2^i) \end{bmatrix} \end{aligned} \quad (3.39)$$

In the case studied, the well known locations and their theoretical configurations are:

$$p_M = \begin{bmatrix} x_M^i \\ y_M^i \end{bmatrix} = \begin{bmatrix} 271.2468 \\ 382.4231 \\ 343.8540 \\ 240.6678 \\ \dots \\ 237.3482 \\ 512.5508 \\ 322.9468 \\ 315.1908 \end{bmatrix}; \quad q = \begin{bmatrix} \theta_1^i \\ \theta_2^i \end{bmatrix} = \begin{bmatrix} 2.0870 \\ -2.3866 \\ 1.7772 \\ -2.4694 \\ \dots \\ 2.1994 \\ -2.2207 \\ 1.9186 \\ -2.4164 \end{bmatrix}; \quad i = 1, \dots, 30 \quad (3.40)$$

Due to the fact that the real robot has small differences with respect to the ideal DH model set into the controller, if q were the commanded values to the real non-calibrated manipulator the real positions achieved would be given by (3.36), but considering the implicit error of (3.37), that is:

$$p_S = \begin{bmatrix} x_S^i \\ y_S^i \end{bmatrix} = \begin{bmatrix} (a_1 + a_{1E}) \cdot c_{1E}^i + a_2 \cdot c_{1E2}^i \\ (a_1 + a_{1E}) \cdot s_{1E}^i + a_2 \cdot s_{1E2}^i \end{bmatrix} \quad (3.41)$$

with

$$\begin{aligned} c_{1E}^i &= \cos(\theta_1^i + \theta_{1E}) & c_{1E2}^i &= c_{1E}^i c_2^i - s_{1E}^i s_2^i \\ s_{1E}^i &= \sin(\theta_1^i + \theta_{1E}) & s_{1E2}^i &= c_{1E}^i s_2^i + s_{1E}^i c_2^i \end{aligned} \quad (3.42)$$

In the case studied, the values reached are:

$$p_S = \begin{bmatrix} x_S^i \\ y_S^i \end{bmatrix} = \begin{bmatrix} 264.4295 \\ 387.2709 \\ 339.5570 \\ 246.8272 \\ \dots \\ 228.2464 \\ 516.7747 \\ 317.3253 \\ 320.9658 \end{bmatrix}; \quad i = 1, \dots, 30 \quad (3.43)$$

The error Δp is calculated like:

$$\Delta p = p_S - p_M = \begin{bmatrix} x^i \\ y^i \end{bmatrix} = \begin{bmatrix} -6.8173 \\ 4.8478 \\ -4.2970 \\ 6.1594 \\ \dots \\ -9.1018 \\ 4.2239 \\ -5.6215 \\ 5.7750 \end{bmatrix}; \quad i = 1, \dots, 30 \quad (3.44)$$

With the Δp data and the DK model of the manipulator (3.36), an iterative procedure is applied. In the example that is being considered, the Jacobian matrix of (3.31) for consecutive s -iterations, $J^{(s)}(q, \beta^{(s)})$, is a (2 x 2) matrix calculated as:

$$J_f = \begin{bmatrix} \frac{\partial x}{\partial a_1} & \frac{\partial x}{\partial \theta_1} \\ \frac{\partial y}{\partial a_1} & \frac{\partial y}{\partial \theta_1} \end{bmatrix}_{\substack{\theta_1 = \theta_1 + \sum_{w=1}^s \Delta \theta_1^{(w)} \\ a_1 = a_1 + \sum_{w=1}^s \Delta a_1^{(w)}}} = \begin{bmatrix} \cos(\theta_1 + \sum_{w=1}^s \Delta \theta_1^{(w)}) & -a_2 \cdot \sin((\theta_1 + \sum_{w=1}^s \Delta \theta_1^{(w)}) + \theta_2) - \sin(\theta_1) \cdot (a_1 + \sum_{w=1}^s \Delta a_1^{(w)}) \\ \sin(\theta_1 + \sum_{w=1}^s \Delta \theta_1^{(w)}) & a_2 \cdot \cos((\theta_1 + \sum_{w=1}^s \Delta \theta_1^{(w)}) + \theta_2) + \cos(\theta_1) \cdot (a_1 + \sum_{w=1}^s \Delta a_1^{(w)}) \end{bmatrix} \quad (3.45)$$

The observation matrix W for the first iteration is composed as (3.27), evaluated at each configuration q^i , considering no-errors at θ_1 and a_1 , that is:

$$\Delta \beta^{(0)} = \begin{bmatrix} \Delta a_1^{(0)} \\ \Delta \theta_1^{(0)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (3.46)$$

It results in a 60x2 matrix, due to the fact that J_f gives a two rows sub-matrix, (3.45), for each observed configuration.

$$W^{(0)} = \begin{bmatrix} J_f(q^1, \beta^{(0)}) \\ \dots \\ J_f(q^{30}, \beta^{(0)}) \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} -0.4936 & -382.4231 \\ 0.8697 & 271.2468 \end{bmatrix} \\ \begin{bmatrix} -0.2049 & -240.6678 \\ 0.9788 & 343.8540 \end{bmatrix} \\ \dots \\ \begin{bmatrix} -0.3408 & -315.1908 \\ 0.9401 & 322.9468 \end{bmatrix} \end{bmatrix} \quad (3.47)$$

then, eq. (3.31) is solved by means of the 2×60 pseudo-inverse of W (3.28),

$$W^{(0)\dagger} = (W^{(0)T} W^{(0)})^{-1} W^{(0)T} = \begin{bmatrix} 0.0117 & -0.0001 \\ 0.0527 & -0.0001 \\ 0.0198 & -0.0001 \\ 0.0523 & -0.0000 \\ \dots \\ 0.0244 & -0.0001 \\ 0.0518 & -0.0001 \\ 0.0174 & -0.0001 \\ 0.0516 & -0.0000 \end{bmatrix}^T \quad (3.48)$$

Finally a first estimation of the error $\Delta\beta$ in both considered DH parameters is obtained

$$\Delta\beta^{(1)} = \begin{bmatrix} \Delta a_1^{(1)} \\ \Delta \theta_1^{(1)} \end{bmatrix} = W^{(0)\dagger}_{2 \times 60} \cdot \Delta p_{60 \times 1} = \begin{bmatrix} 0.0269 \\ 0.0177 \end{bmatrix} \quad (3.49)$$

The following step for this calibration consists of actualizing the DK model parameters by adding the obtained $\Delta\beta^{(1)}$ to the ideal DH parameters $\beta^{(0)}$. Table 3.2 shows the modifications. The new parameters will be referred as $\beta^{(1)}$:

i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
1	0	$a_1 + \Delta a_1^{(1)} = 650.0269$	$\theta_1 + \Delta \theta_1^{(1)} = \theta_1 + 0.0177$	0
2	0	$a_2 = \sqrt{600^2 + 155^2}$	θ_2	0

Table 3.2. MDH Matrix simulating the manipulator gross positioning.

The DK model considered to actualize the p_M positions achieved with these corrections, for the s -iteration, is

$$p_M^{(s)} = \begin{bmatrix} x_M^i \\ y_M^i \end{bmatrix}^{(s)} = \begin{bmatrix} (a_1 + \sum_{w=1}^s \Delta a_1^{(w)}) \cdot c_{\Delta 1}^i + a_2 \cdot c_{\Delta 12}^i \\ (a_1 + \sum_{w=1}^s \Delta a_1^{(w)}) \cdot s_{\Delta 1}^i + a_2 \cdot s_{\Delta 12}^i \end{bmatrix} \quad (3.50)$$

with

$$\begin{aligned} c_{\Delta 1}^i &= \cos(\theta_1^i + \sum_{w=1}^s \Delta \theta_1^{(w)}) & c_{\Delta 12}^i &= c_{\Delta 1}^i c_2^i - s_{\Delta 1}^i s_2^i \\ s_{\Delta 1}^i &= \sin(\theta_1^i + \sum_{w=1}^s \Delta \theta_1^{(w)}) & s_{\Delta 12}^i &= c_{\Delta 1}^i s_2^i + s_{\Delta 1}^i c_2^i \end{aligned} \quad (3.51)$$

After actualizing the actual error Δp as in (3.44), a new observation matrix $W^{(1)}(q, \beta^{(1)})$ is obtained with (3.45) like in (3.47).

Reconsidering (3.49), a new correction $\Delta \beta^{(2)}$ is obtained.

$$\Delta \beta^{(2)} = \begin{bmatrix} \Delta a_1^{(2)} \\ \Delta \theta_1^{(2)} \end{bmatrix} = W_{2 \times 60}^\dagger \cdot \Delta p_{60 \times 1} = \begin{bmatrix} 0.1732 \\ -0.0002 \end{bmatrix} \quad (3.52)$$

It can be appreciated that, in the second iteration, the error introduced (3.37) is reached, that is, $\Delta \beta^{(1)} + \Delta \beta^{(2)} = \Delta \beta_E$. Consequently, in the following iterations, the corrections $\Delta \beta^{(s)}$ obtained are almost null, verifying (3.30) so the iterative process would end. Table 3.3 shows the final model achieved with the corrections done:

i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
1	0	$a_1 + \Delta a_1^{(1)} + \Delta a_1^{(2)} = 650.2$	$\theta_1 + \Delta \theta_1^{(1)} + \Delta \theta_1^{(2)} = \theta_1 + 0.0175$	0
2	0	$a_2 = \sqrt{600^2 + 155^2}$	θ_2	0

Table 3.3. MDH Matrix simulating the manipulator gross positioning.

3.3. CALIBRATION OF THE ADDITIONAL JOINTS OF THE KUKA WORKCELL

The main problem in many traditional calibration methods is the need to have an accurate, fast and not expensive equipment to measure the EE's pose. It also has been exposed that autonomous calibrations methods are possible for robot manipulators with either some a priori knowledge of the task constraint or redundancy of the sensing systems (e.g., planar constraints).

For this particular case, the *open-loop* method presented uses a set of positions of the terminal point of the robot which are assumed to be in the same plane, but avoiding any physical contact by using a laser displacement sensor. This technique arises from the fact that the use of a contact probe is problematic, because it is difficult to be certain that the tip-point of the probe is exactly on the surface.

In this *Non-contact Planar Constraint Calibration procedure*, the laser sensor sweeps three orthogonal constraint planes (namely, a squared pattern) set in its workspace while measuring the distance. Only the distance readings are recorded. A NLSQ identification model has been derived from the consistency conditions of the planes, and is detailed below. The proposed method can be applied to the full-articulated chain by magnifying the observed parameters with the same guidelines and it is suitable for on-site calibration in an *industrial environment* at regular intervals. This calibration scheme can be implemented to be done autonomously.

In this particular case, the method is implemented for the calibration of the external additional joints introduced in Chapter 2, namely the linear track d_L and the rotary table θ_M , due to the fact that the assembling of the workcell *in situ* carried out by the operators left some misalignments while the manipulator has good accuracy itself. Nevertheless, the method can be applied to the full articulated chain by magnifying the Jacobian matrix.

3.3.1. *Non-contact Planar Constraint Calibration procedure. Material and method.*

A *planar constraint* method for robot calibration may be classified into *single-plane* and *multiple-plane* constraints. Zhuang et al. [22] showed that whenever a single plane constrains the robot motion, the calibration result is biased because the measurement data is projected to a particular direction. Furthermore, a single-plane constraint does not necessarily guarantee the observability of unknown kinematic parameters of the robot. These authors demonstrated that, if measurements are constrained to lie on three mutually non-parallel planes, data collected by this *multiple-plane constraint setup* is equivalent to that by a point measurement device. The significance of this observation is that a 3D-position measurement system may be replaced by no less than three planes placed at a number of different orientations. Without loss of generality and in practical terms of *industrial calibration*, the calibration issue can be carried out on a mechanized squared corner (Figure 3.7) placed on the base frame of the workspace $\{B\}$ in which accuracy is critical (i.e., it will be assumed that the plane parameters are known *a priori*).

At this point, some authors [3][23] work by means of recording the joint readings enabling desirable and safe touch on the planes. Actually, the use of a contact probe is problematic because it is difficult to be certain that the tip-point of the probe is exactly on the surface, neither above it nor indenting it. In the scheme presented, a *laser displacement sensor* attached to the EE aims at arbitrary but fixed points on three orthogonal meshes placed in those three planes.

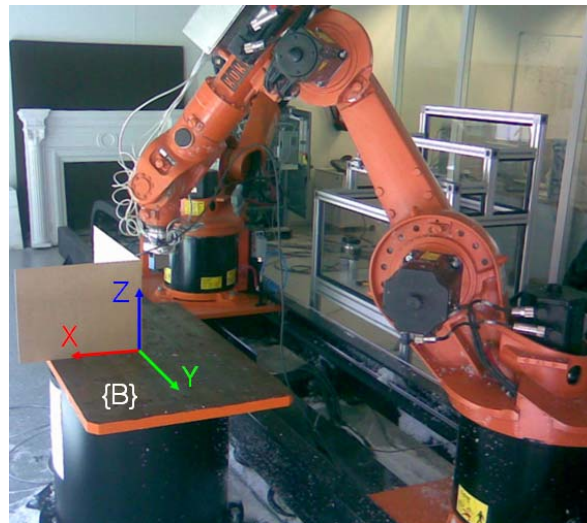


Figure 3.7. Mechanized squared corner with high tolerance degree, placed on the base frame of the workspace $\{B\}$

The procedure capitalizes on the constraint that the laser line, being almost perpendicular to the plane, must be at a constant distance of each plane for all the respective points of the mesh commanded to the controller. The main advantage of the scheme presented in this Section is that the distant laser point is very sensitive and there is no physical contact, which facilitates acquiring more accurate readings for the calibration. Instead of working with the joint readings, the NLSQ identification model is derived only from the laser readings without external measures or joint recordings, see Figure 3.8.

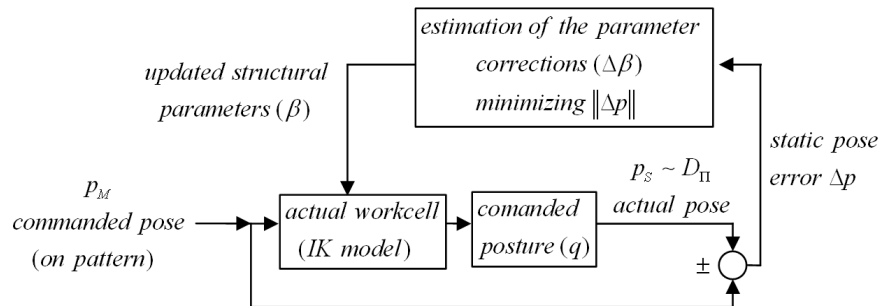


Figure 3.8. Open-loop procedure proposed for the workcell calibration.

i) Laser displacement sensor

A laser displacement sensor (mod. SICK OD100-35P840) is rigidly attached to the robot flange by means of a specific tool holder. It has a measuring range of 100 mm with 35 μm in resolution. The laser is supposed to be aligned with the Z-axis of the EE and a coordinate system ($\{\text{LR}\}$) is chosen in the laser line. Both X-Y axes orientation and the origin (TCP) along the laser line are arbitrary, which can be set at some convenient location. As it is depicted at Figure 3.9 (right), the triangulation measurement is the physical basis of this displacement sensor: the site of the light spot on the *position-sensitive device* (PSD, a photodiode) is dependent on the distance of the detected object. The signals A and B (see Figure 3.9) change depending on the position of the light spot. The calculation of the signals in the microcontroller then gives a linear output signal depending on the distance of the object.

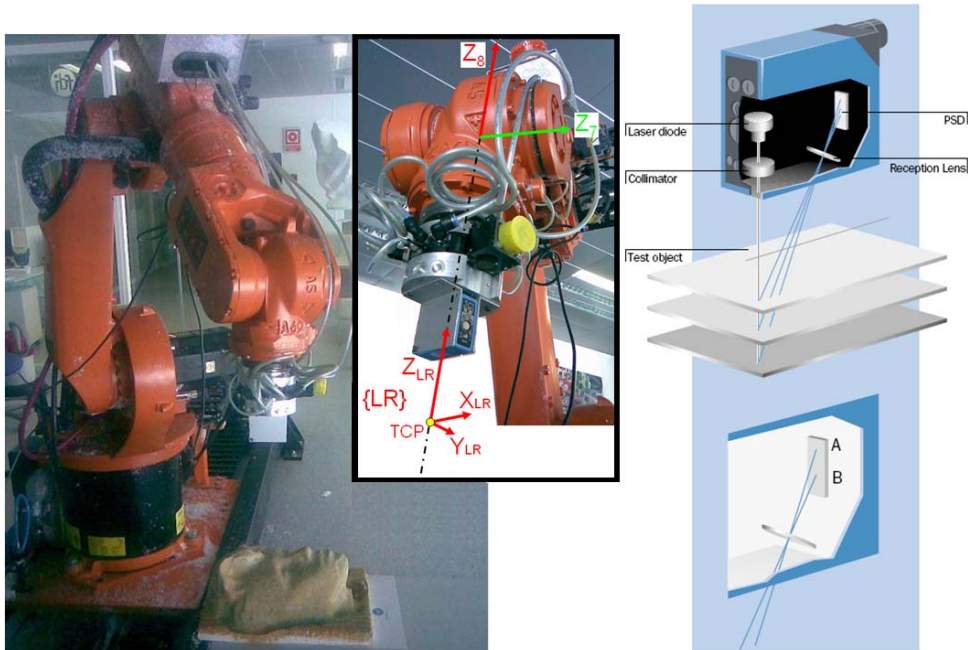


Figure 3.9. View of the laser displacement sensor.

The laser is supposed to be aligned with the Z_8 -axis of the EE and a coordinate system ($SC\{LR\}$) is chosen in the laser line, see Figure 3.9 and Figure 3.10, left. Both the orientation of X-Y axes and the origin along the laser line are arbitrary. Then, the robot can aim the laser TCP at some location.

It may be cumbersome to use *planar methods* if the calibration process includes the calibration of the laser itself. The observability of certain parameters is not possible with the EE being moved parallel over a plane as described. These unknown parameters are: two rotations about X_8 and Y_8 (these angles of rotation are close to zeros) and two coordinates of translation in X_8 - Y_8 plane (also close to zeros). Gatla et al. [5] integrate this calibration procedure into the whole calibration process, but in a different way the joint values are not recorded in the method proposed here. Zhu et al. [26] also isolate this problem and proposed an approach for calibrating robot tool center point (TCP) position relative to the robot-mounting flange when using a noncontact sensor (such as the laser pointer tool) through the use of NLSQ optimization algorithms and simple geometry with known dimensions. Thus, we assume the availability of a fully calibrated laser sensor prior to the assembly of the new additional joints, i.e. ${}^8T_{LR}$ is known.

3.3.2. Formulations

i) Formulation of the Kinematic Identification Model

The *planar methods* use a set of configurations of the manipulator where the position of the terminal point (the TCP) of the robot is in the same plane. In the Cartesian workspace $\{B\}$, the terminal points are commanded to the controller by setting the position and orientation of the TCP. For each commanded point,

$$p_M^i = (p_{Mx}^i, p_{My}^i, p_{Mz}^i, p_{MA}^i, p_{MB}^i, p_{MC}^i) = f(q^i, \beta); \quad i = 1, \dots, np \quad (3.53)$$

where np is the total number of points and $p_{Mx}^i, p_{My}^i, p_{Mz}^i$ represent the position in $\{B\}$, whereas the orientation is set by three RPY² angles, $p_{MA}^i, p_{MB}^i, p_{MC}^i$. The mp adjustable model parameters are held in the vector β . In the case studied, $\beta = [\alpha_1 \ a_1 \ \theta_M \ d_1 \ \theta_2 \ d_L]^T$ ($mp=6$).

The general equation of a plane Π containing the origin of $\{B\}$ is:

$$a x + b y + c z = 0 \quad (3.54)$$

where a, b, c correspond to the plane coefficients. Since the TCP of the laser is supposed to be in the plane, each commanded point p_M^i should accomplish:

$$a p_{Mx}^i + b p_{My}^i + c p_{Mz}^i = 0 \quad (3.55)$$

Equation (3.55) is exploited to carry out the calibration of the robot parameters. In practical terms of *industrial calibration*, it is useful to assume that the coefficients of the plane are known because the working space is well known and $\{B\}$ is already identified in it. This base $\{B\}$, materialized with a squared pattern, can be placed easily using an external sensor where a very limited number of points is needed (theoretically only three points per plane are needed), see Figure 3.7. Moreover, the particular equations observed for the three pattern planes are, respectively:

$$p_{Mx}^i = 0 \quad (3.56)$$

$$p_{My}^i = 0 \quad (3.57)$$

² RPY (roll-pitch-yaw) KUKA convention [34].

$$p_{Mz}^i = 0 \quad (3.58)$$

We name these planes respectively as Π_x, Π_y and Π_z . For each plane, a set of points in those planes can be easily done by commanding the robot controller the location of three different orthogonal meshes (Figure 3.10, right), in which the corresponding coordinate (p_{Mx}^i, p_{My}^i or p_{Mz}^i) is null and the orientation of the laser pointer is perpendicular to the plane, respectively:

$$p_M^i = (0, p_{My}^i, p_{Mz}^i, 0, -\pi/2, 0) \quad (3.59)$$

$$p_M^i = (p_{Mx}^i, 0, p_{Mz}^i, 0, 0, -\pi/2) \quad (3.60)$$

$$p_M^i = (p_{Mx}^i, p_{My}^i, 0, 0, 0, 0) \quad (3.61)$$

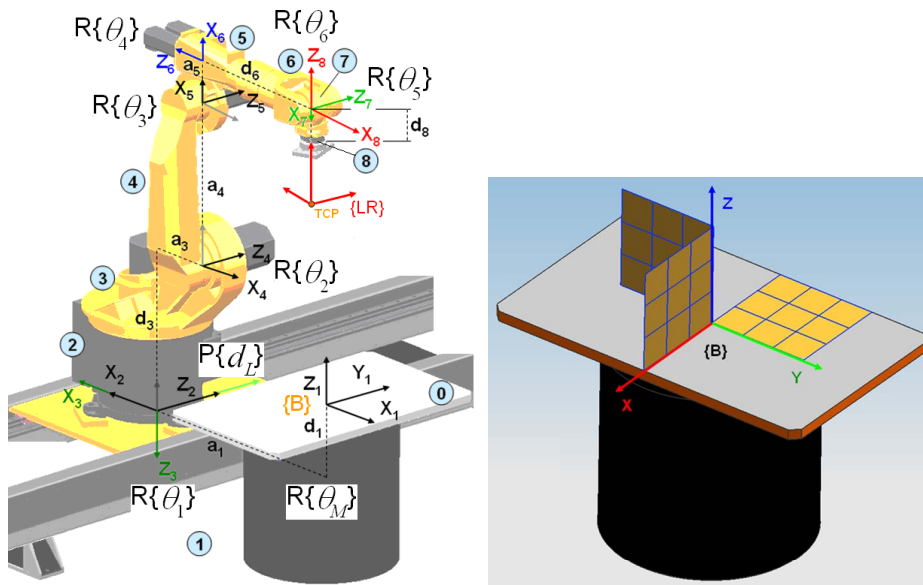


Figure 3.10. Left, highlight of the DH model introduced in Chapter 2. Right, three commanded meshes on the three respective planes.

The fact that each set is constrained to lie on its corresponding plane leads to the construction of the identification model. Let p_s^i represent the actual coordinates of the laser TCP in $\{B\}$ that correspond to the configuration q^i acquired for a commanded p_M^i . Due to the perpendicular orientation commanded, the distance D_{Π}^i measured by the

laser device approximates to the respective actual coordinate $p_{S_x}^i$, $p_{S_y}^i$ or $p_{S_z}^i$ at $\{B\}$ (Figure 3.10). For each plane, the other five coordinates are neither unknown nor approximated with any external measure.

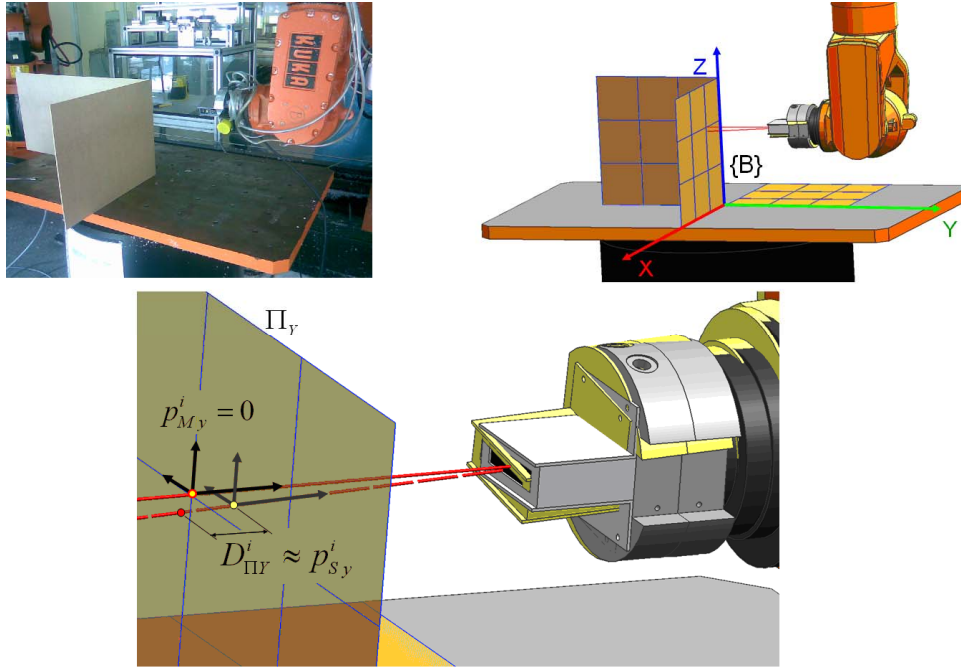


Figure 3.11. Coordinate $p_{S_y}^i$ approximated by the distance $D_{\Pi Y}^i$ to Π_Y .

With this suppose, a residual, r^i , can be defined as the difference between the actual observed values p_S^i and the model predicted values, namely $r^i = p_S^i - f(q^i, \beta)$. For each plane Π and each commanded i -point, we approximate each residual to be minimized with the measure D_{Π}^i . Thus, the identification model relating the deviation of the EE location from the plane Π with the differential error in the geometric parameters ($\Delta\beta$) can be expressed as:

$$D_{\Pi} = J_{f_{\Pi}} \Delta\beta \quad (3.62)$$

with $J_{f_{\Pi}} = \partial r_{\Pi} / \partial \beta$ being the *Jacobian* matrix of the residual regarding the identifiable parameters. Reasonably, only the expression corresponding to

the residual error in the perpendicular direction of the plane Π , r_{Π} , is taken into account.

In this non-linear system, the derivatives $\partial r_{\Pi} / \partial \beta$ are functions of both the commanded postures and the identifiable parameters, so these gradient equations do not have a closed solution. Instead, the default values from the operator's assembly are chosen as initial guess for the mp parameters, i.e. $\beta^{(0)}$. Then, while minimizing the residuals, the final value of β is refined iteratively by consecutive approximations,

$$\Delta \beta = \beta^{(s+1)} - \beta^{(s)} \rightarrow \beta^{(s+1)} = \beta^{(s)} + \Delta \beta \quad (3.63)$$

where s is an iteration number. As equation (3.62) is applied for a sufficient number $np \gg mp$ of commanded points arranged in the three orthogonal planes, exceeding the total number of parameters, the resulting system to identify $\Delta \beta$ is

$$\Delta p^{(s)} = W(\beta^{(s)}) \Delta \beta \quad (3.64)$$

where W is the *Observation Matrix* of dimension $np \times mp$. It is an ordered composition of the Jacobians associated to the corresponding observation at each plane, $W = [J_{f_{\Pi_x}}(\beta^{(s)}) \ J_{f_{\Pi_y}}(\beta^{(s)}) \ J_{f_{\Pi_z}}(\beta^{(s)})]^T$. Consequently, care must be taken in the configuration of Δp due to the fact that the measures taken in each plane must correspond with the significance of each row of W . The $np \times 1$ vector of the observed residuals in the three planes is $\Delta p^{(s)} = [D_{\Pi_x}^{(s)} \ D_{\Pi_y}^{(s)} \ D_{\Pi_z}^{(s)}]^T$.

As in previous sections, the increment $\Delta \beta$ can be solved to get the least-squares errors solution,

$$\Delta \beta = W^{\dagger} \Delta p^{(s)} \quad (3.65)$$

where W^{\dagger} is the *left pseudo-inverse* of W , namely $W^{\dagger} \equiv (W^T W)^{-1} W^T$. The geometric parameters β are iteratively updated in Δp and W^{\dagger} , eq. (3.63), until the elements of $\Delta \beta$ become smaller than some prescribed limit (see eq. (3.30)). This best-fit solution is kept in the controller.

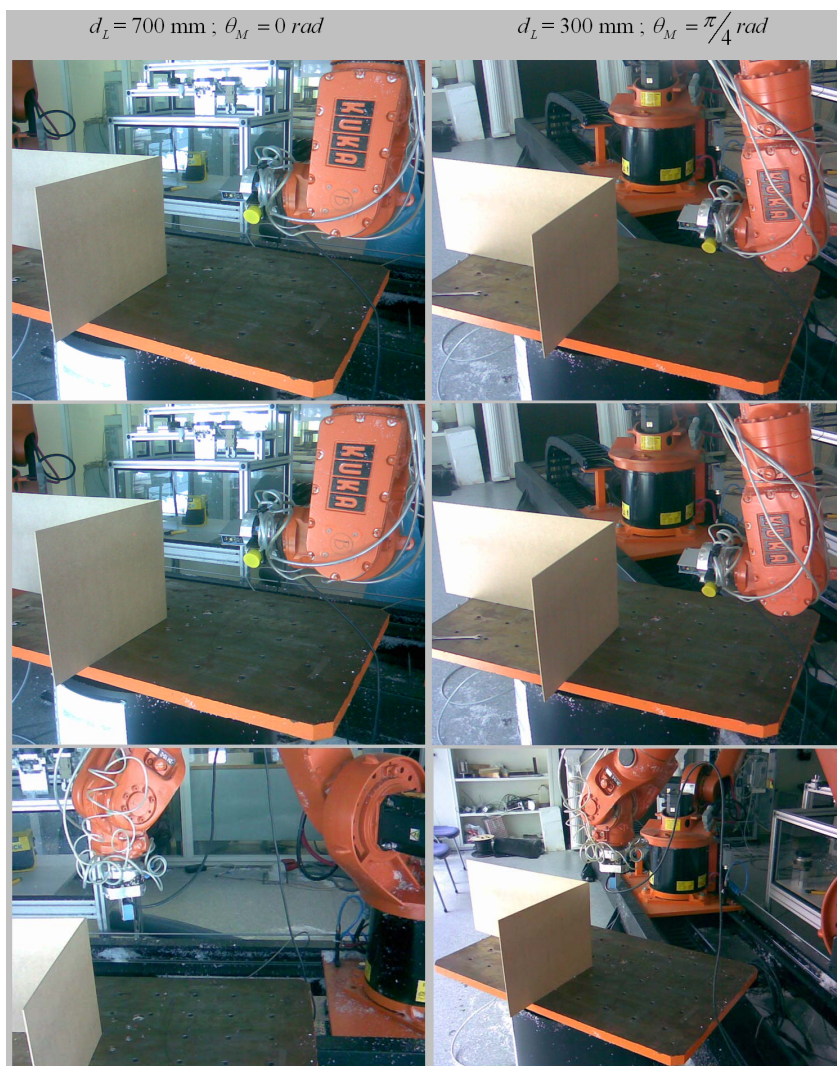


Figure 3.12. Sweeping the reference planes with two configurations of d_L and θ_M .

3.3.3. Results

The calibration procedure was run in an Intel™ Core Duo PC with Matlab™ 2007c. It showed a good convergence for the studied workcell, with final values accomplishing the convergence criterion at the 18th iteration. The final corrections achieved were $\Delta\beta = [0.01 \ 0.06 \ 0.07 \ 0.05 \ 0.01 \ 0.08]^T$ (mm, rad)

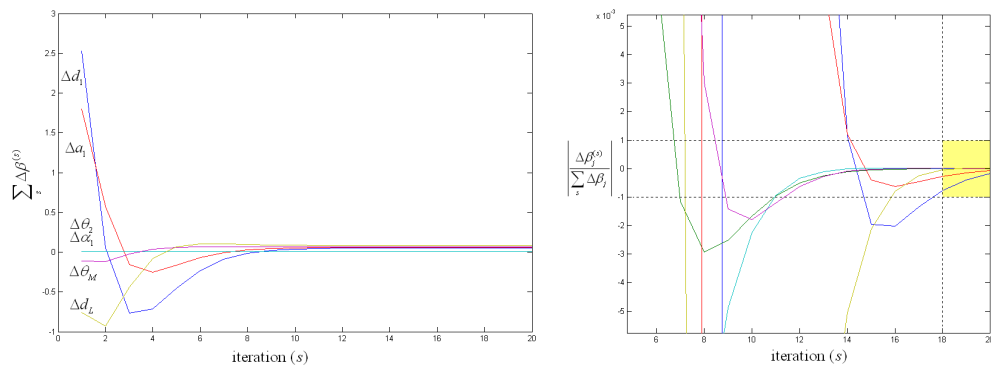


Figure 3.13. Left, the values of the increment $\Delta \beta$ show a final stable value; Right, the stop criterion is achieved after 18th iterations.

REFERENCES (Ch. 3)

- [1] International Standard ISO 9283, "Manipulating industrial Robots: Performance criteria and related test methods".
- [2] M. Abderrahim, A. Khamis, S. Garrido, L. Moreno; "Industrial Robotics: Programming, Simulation and Applications", Chap. 7: "Accuracy and Calibration Issues of Industrial Manipulators", ARS/pIV, Germany, December 2006. ISBN 3-86611-286-6
- [3] X.-L. Zhong and J. M. Lewis, "A New Method for Autonomous Robot Calibration", IEEE Int. Conf. on Robotics and Automation, pp. 1790-1795, ISSN 0-7803-1965-6 (1995)
- [4] Mooring, B. W., Z. S. Roth, and M. Driels, "Fundamentals of manipulator calibration", pp 23-48, John Wiley & sons, Inc.
- [5] Ch. S. Gatla, R. Lumia, J. Wood, and G. Starr; "An Automated Method to Calibrate Industrial Robots Using a Virtual Closed Kinematic Chain"; IEEE Transactions on Robotics, Vol. 23, No. 6, pp. 1105-1116, December 2007
- [6] I. Fassi, G. Legnani, D. Tosi, A. Omodei; "Industrial Robotics: Programming, Simulation and Applications", Chap. 8: "Calibration of Serial Manipulators: Theory and Applications", ARS/pIV, Germany, December 2006. ISBN 3-86611-286-6
- [7] Whitney, D., Lozinsky and Rourke, J., "Industrial robot forward calibration method and results", ASME J. of Dynamic Systems, Measur. and Control, 108, pp. 1-8, 1986.
- [8] M. R. Driels, L. W. Swayze, and L. S. Potter, "Full-pose calibration of a robot manipulator using a coordinate measuring machine," Int. J. Adv.Manuf. Technol., vol. 8, no. 1, pp. 34–41, 1993.
- [9] Zhong, XL, Lewis, J.M., and Rea, H.; "Neuroaccuracy Compensator for Industrial Robots"; Proc. Of IEEE Int. Conf. on Neural Networks, WCC1'94, Vol. 5, pp. 2797-2802, Orlando, Florida, 1994
- [10] Stone, H.W., Sanderson, A.C. and Neuman, C.P., "Arm Signature Identification", *Proc. IEEE Int. Conf. On Robotics and Automation*, San Francisco, CA, PP. 41-48, 1986.
- [11] Driels, M.R. and Swayze, W., "Automated partial pose measurement system for manipulator calibration experiments", *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 4, PP. 430-440, 1994.
- [12] J.R. Prenninger, M. Vincze, H. Gander; "Contactless position and orientation measurement of robot EE", *IEEE Int. Conf. Robotics and Automation*, Atlanta, Vol.1: pp. 180-185, 1993
- [13] M. Vincze, J. P. Prenninger, and H. Gander, "A laser tracking system to measure position and orientation of robot EEs under motion", Int. J. Robot. Res., vol. 13, pp. 305–314, 1994.
- [14] D.W. Osborn and W. S. Newman, "A new method for kinematic parameter calibration via laser line," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1993, vol. 2, pp. 160–165.

- [15] L. Giugovaz and J. M. Hollerbach, "Closed loop kinematic calibration of the Sarcos Dexterous Arm," in Proc. IEEE/RSJ Intl. Conf. Intell. Robots Syst. Sep. 12–16, 1994, pp. 329–334.
- [16] D. J. Bennet and J. M. Hollerbach, "Autonomous calibration, of singleloop closed kinematic chains, formed by manipulators with passive endpoint constraints", IEEE Trans. Robot. Autom., vol. 7, no. 5, pp. 597–606, Oct. 1991.
- [17] M. A. Meggiolaro, G. Scriffignano and S. Dubowsky; "Manipulator calibration using a single endpoint contact constraint"; Proceedings of the 26th Biennial Mechanisms and Robotics Conference of the 2000 ASME Design Engineering Technical Conferences, Baltimore, MD, September 2000.
- [18] W. Khalil, G. Garcia, J.F. Delagarde, "Calibration of the Geometric Parameters of Robots without External Sensors", IEEE Int. Conf. on Robotics and Automotion, pp. 3039-3044, ISSN 0-7803-1965-6 (1995)
- [19] W. Khalil, P. Lemoine; "GECARO: A System for the Geometric Calibration of Robots", APII-JESA European Journal of Automation, Publisher Hermes Science Publications, ISSN 1269-6935 33, 5-6 (1999) pp. 717-739
- [20] W. Khalil, S. Besnard, P. Lemoine, "Comparison Study of the Geometric Parameter Calibration Methods", International Journal of Robotics and Automation 15, 2 (2000) pp. 56-67
- [21] Ikits, M. and J. Hollerbach, "Kinematic calibration using a plane constraint", Proc. IEEE Int. Conf. Robotics and Automation, pp. 3191-3196, April 20-25, 1997.
- [22] H. Zhuang, S. H. Motaghedi, and Z. S. Roth, "Robot calibration with planar constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, Detroit, MI, 1999, pp. 805–810.
- [23] W. Khalil, P. Lemoine, M. Gautier; "Autonomous calibration of robots using planar points", International Symposium on Robotics and Manufacturing-ISRAM'96, Montpellier, France (1996)
- [24] J. Hollerbach; "A review of kinematic calibration", *The Robotics, Review 1* (Cambridge, MA: MIT Press, 1989), pp. 207-242.
- [25] P. Robinson, P. Orzechowski, P.W. James, C. Smith, "An Automated Robot Calibration System", IEEE-Proceedings ISIE'97, pp. 285-290 IEEE, Catalog Number: 97/THS280, Guimariies, Portugal (1997).
- [26] Z. Zhu, Q. Tang, J. Li, Z. Gan ; "Calibration of laser displacement sensor used by industrial robots"; Society of Photo-Optical Instrumentation Engineers, Opt. Eng. 43(1) pp. 12–13, DOI: 10.1117/1.1631935, January 2004.
- [27] K.S. Fu, R.C. Gonzalez and C.S.G. Lee. (1987). *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw Hill, Singapore.
- [28] J. J. Craig, *Introduction to Robotics*. Reading, MA: Addison-Wesley, 1986.
- [29] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *ASME J. Appl. Mech.*, vol. 77, pp. 215–221, 1955.
- [30] H. Zhuang, Z.S. Roth and F. Hamano. 1992. A Complete and Parametrically Continuous Kinematic Model for Robot Manipulators. *IEEE Transactions on Robotics and Automation* , Vol.8, No.4.
- [31] S.A. Hayati and M. Mirmirani. 1985. Improving the absolute positioning accuracy of robot manipulators. *J. Robotic Systems*. 2: 397-413.
- [32] Hollerbach, J.M. and C.W. Wampler. (1996). A taxonomy of kinematic calibration methods. *International Journal of Robotics Research*, 14, pp. 573-591.

[33] http://en.wikipedia.org/wiki/Moore%E2%80%93Penrose_pseudoinverse

[34] KUKA Corp., 2005. KUKA System Software (KSS): Expert Programming (KRC2 / KRC3), Release 5.2.

CHAPTER 4

CAM TO WORKCELL POSTPROCESSING

"One of the principal objects of theoretical research in my department of knowledge is to find the point of view from which the subject appears in its greatest simplicity." –

J. W. Gibbs

CHAPTER 4. CAM TO WORKCELL POSTPROCESSING

4.1. INTEGRATED PRODUCTION SYSTEMS

In the latter years, a radical change has taken place in the design and manufacture of all industrial products. The CAD/CAM/CNC-ROB systems, which integrate the labours of piece design and generation of paths for its machining, are included into a wider concept of *Computer Integrated Manufacturing (CIM)* (Figure 4.1).

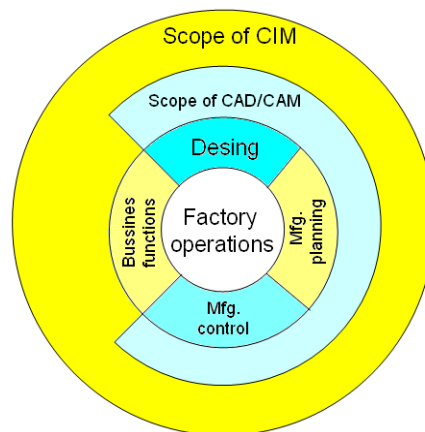


Figure 4.1. The scope of CAD/CAM and CIM [2]

CAD stands for *Computer-Aided Design*. These systems are meant for engineering drafting and drawing based mostly on lines, arcs, spline curves and, more recently, 3-D surfaces. Packages such as AutoCAD™, NX™, SolidWorks™, SolidEdge™, Catia™ and ProEngineer™ are some of the more commonly used CAD systems on the market today.

CAM systems (*Computer-Aided Manufacturing*) take the drawing to the final stage to produce machining instructions to make the part on a CNC (*Computer Numerically Controlled*) machine (milling machine, lathe, but also including manipulators as it is the scope of this thesis).

Historically, CAD systems began as a technological computerized engineering, while the CAM were a semiautomatic technology for the control of machines of numerical form [1]. But these two disciplines, which were born separately, have been mixed gradually up to obtaining a technology sum of the

two, so that CAD/CAM systems are considered to be a unique discipline nowadays, see Figure 4.2. This means that the separation between design and manufacture has diminished to the minimum and the qualification of the technical staff has improved substantially with regard to a few years ago. The design process has changed itself and nowadays it is done by means of an interactive dialog between the designer and the computer. In this sense, the designer is released of the least creative processes to intensify his effort in those tasks that cannot automate (like the creative processes). It is important to note that the quality and efficiency of the CAD-models designed is the point of departure for applying the rest of computer assisted technologies, see Figure 4.3.

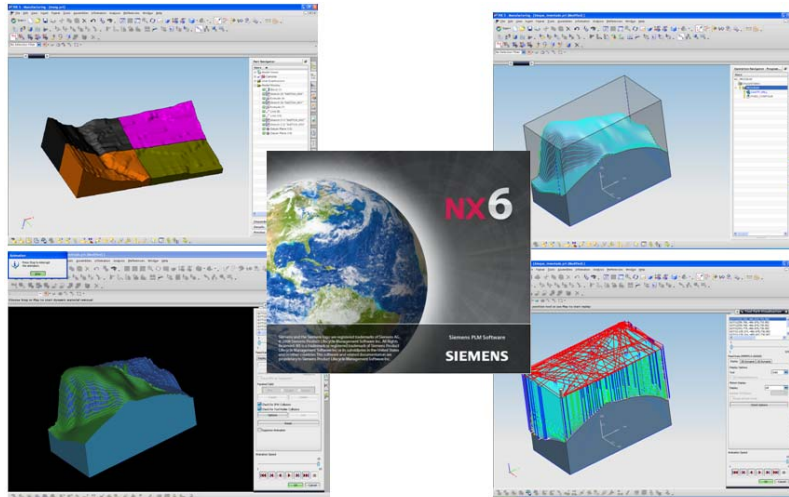


Figure 4.2. View of the main window of NXTM while covering a complete CAD/CAM process.

Due to particular characteristics of each manufacturing CNC-machine (kinematics, controller languages, etc), an important step has to be done between CAD/CAM and the final production stage, namely, the *postprocessing* to the specific machine to be used. This concept is discussed in Section 4.4.

4.1.1. Benefits of the integrated production systems

The benefits in the adoption of integrated production systems can be summarized in [1][2]:

- Ability to handle pieces with increased complexity,

- Notable increase of the productivity: reduction in errors, elimination of dry runs and time saving,
- Decrease of the production costs,
- Rapid adaptability of the production to the fluctuations and requirements of the market,
- Considerable improvement in quality and reliability of the production,
- Professional promotion of the technicians and skilled workers, and more effective utilization of the machinery.

The *simplicity* in the *flow of information* between the production stages (Figure 4.3) is the most significant contribution of the CAD/CAM integration. In one hand, it implies that the information relating to the manufacturing process, to the quality or to the costs of the product becomes more visible to the engineering (CAD). On the other hand, it also implies that the information of the design of any product is more accessible when aiming for the manufacturing path-planning (CAM).

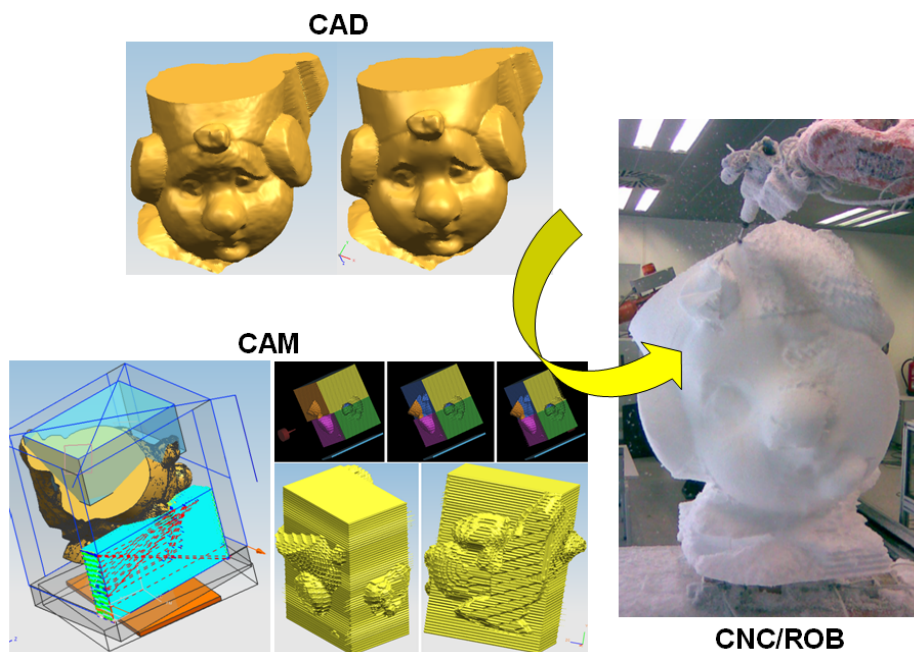


Figure 4.3. CAD/CAM/CNC-ROB flow process: the quality of the CAD-model determines the efficiency of the results obtained in the following steps of the manufacturing process.

However, it seems to be that the direction of development of CAD/CAM systems is directed towards engineers rather than machinists. From the author's point of view, and as it will be shown in following Sections, in some systems skill-based knowledge is not given the emphasis that it rightly deserves. The adaptation of the information to CNC-machines, including robots (ROB), makes more evident the feedback needed between the machinists and engineers.

4.2. COMPUTER NUMERICAL CONTROL (CNC)

Machine tools have played a fundamental role in the technological development of the World up to the point that the rate of development of the machines tools is directly related to the rate of the industrial development. To a certain extent, it is what happened at the Industrial Revolution with the vapour machine.

Day after day, numerous and new requirements forced the utilization of new computerized technologies partially replacing the human operator. About 1942, the engineer John T. Parsons come up what might be called the first *numerical control* (NC), due to a need imposed by the aeronautical industry for the accomplishment of propellers of helicopters with different configurations. Since then, the Computer Numerical Control (CNC) has taken its place in industry for several reasons [3]:

- Need to make products that could not be obtained in sufficient quantity and quality (precision) without an automation of the manufacturing process,
- Need to obtain products very difficult to make or even impossible until then, for being excessively complex to be controlled by a human operator (see Figure 4.4). Even in case of small series the use of NC can be profitable when the piece is complex enough to justify its programming,
- Need to make products with low prices,
- Security: NC is especially advisable for the work with dangerous products.

Later, due to the new needs in industry other important factors appeared such as the flexibility, which also promoted the use of robotic manipulators (ROB).

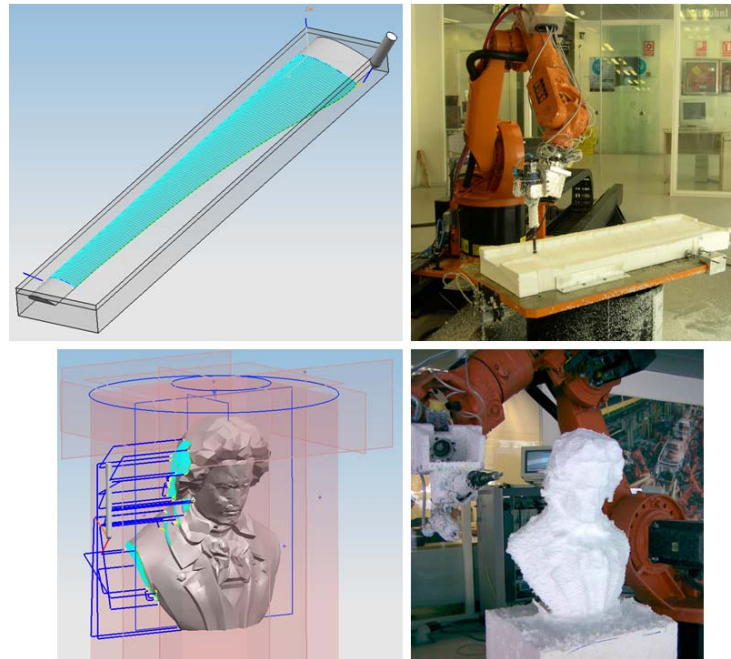


Figure 4.4. CAD/CAM/ROB systems offer the possibility of producing very complex pieces

4.2.1. Definition

CNC (Computer Numerical Control) refers specifically to *the method of controlling machines by the application of digital electronic computers and circuitry* [4]. Machine movements that are controlled by cams, gears, levers, or screws in conventional machines are directed by computers and digital circuitry in CNC-machines from a prepared program containing coded alphanumeric data. Thus, CNC-machine tools or manipulators are operated by programmed commands encoded on a storage medium, as opposed to manually controlled via handwheels or levers, or mechanically automated via cams alone.

4.2.2. Classification of the Numerical Control systems

Generally, NC systems can be classified into:

- Equipments of positioning numerical control or point-to-point (PTP).
- Equipments of contouring (or *continuous path*, CP) numerical control.

In a PTP system, the control determines, from the information given by the program and before the movement, the complete distance to cover. Later the above mentioned positioning is done, without considering the crossed path, since the only thing that matters is to reach with accuracy and rapidity the point commanded.

The equipments that allow generating curves receive the name of *contouring systems*. These machines govern not only the final position but also the movement in every instant of the axes in which the interpolation is realized. In these equipments a perfect synchronization exists among the different joints controlling, therefore, the CP that the tool must follow.

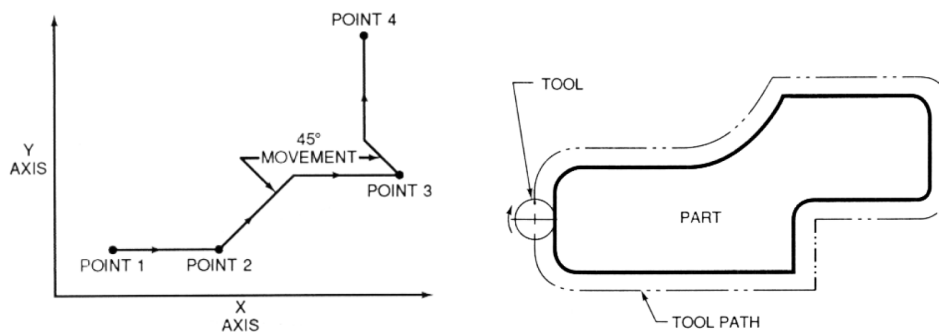


Figure 4.5. Left, the path followed by PTP positioning to reach various programmed points (machining locations) on the XY axis. Right, complex contour tracking.

While the PTP systems are conceived for tasks such as point welding or pick-and-place, more complex tracking can be generated with the contouring systems such as straight lines with any slope, arcs of circumference or any combination of them. These systems are used, especially, for complex milling operations, lathe, etc.

Originally this differentiation referred to different types of machine-tools, since industrial robots were devoted to PTP operations. The evolution in recent years in informatics allowed the design of controllers for more complex robotic manipulators (Figure 4.6).

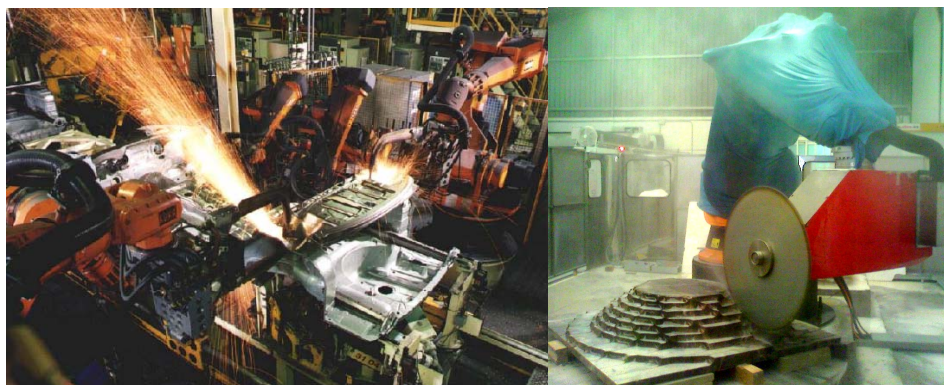


Figure 4.6. Left, resistance welding in the industry of automotion (PTP operation). Right, continuous path (CP) tracking to cut a stone (courtesy of *Pedra Navas*).

i) Interpolation

The method by which contouring machine-tools move from one programmed point to the next is known as *interpolation*. This ability to merge individual axis points into a predefined toolpath is built into most of today's control units. There are five common methods of interpolation, namely: *linear*, *circular*, *helical*, *parabolic*, and *cubic*.

In a *linear interpolation*, the end point of one segment becomes the start point for the next segment, and so on, throughout the entire program. Therefore, a very large number of points would have to be programmed to describe the curve in order to produce a contour shape. *Circular interpolation* has greatly simplified the process of programming arcs and circles. To program an arc, the control unit only requires the coordinate positions of the circle center, the radius of the circle, the start point and end point of the arc being cut, and the direction in which the arc is to be cut (clockwise or counterclockwise). The information required may vary with different controllers.

All contouring controls provide linear interpolation, and most controls are capable of both linear and circular interpolation, such as in the controllers of the robots used for the purpose of this thesis (KUKA™ robots, as those on Figure 4.6). Helical, parabolic, and cubic interpolation are used by industries that manufacture parts which have complex shapes, such as aerospace parts and dies for car bodies, by means of very specific machine-tools.

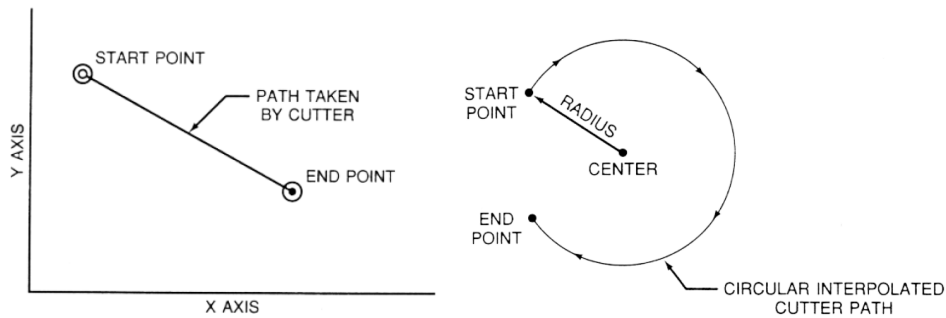


Figure 4.7. Compared with a *linear* controller (left), *circular interpolation* has greatly simplified the process of programming arcs and circles, and consequently the length of the codes.

4.3. CAM SYSTEMS FOR TOOLPATH GENERATION

Once the part to be machined is represented by a CAD model, a set of machining instructions must be produced. These instructions are needed to guide the cutter TCP (*tool center point*) on the path over the raw material.

CAM systems work in three stages [8]: the first is to generate cutter contact points (CC points), the second stage is to generate *cutter location* data (CL-data) and the last stage is to convert CL-data to machine code, for the desired CNC. CL-data contains the necessary information to generate, through any specific *postprocessors*, the numerical commands to drive any specific machine-tool.

Traditionally, CAM has been considered as a numerical control programming tool, wherein CAD models are used to generate the code to drive CNC machine tools. Commercial CAM systems are applied off-line, that is, previously and independently from the machine tool which will manufacture the object. These systems do not eliminate the need for skilled professionals. In fact, leverages the value of the most skilled manufacturing professionals through advanced productivity tools. Users must select some input parameters such as the type of tool, machining process, tolerances, materials and strategies to be used, prior to the generation of the toolpath on the CAD part (Figure 4.8)

4.4. POSTPROCESSING

The latest commercial CAD/CAM systems can design freeform surfaces and generate either the three-axis or five-axis toolpaths. The CL-data, composed of the cutter tip (TCP) *position* and *orientation* relative to the *part frame* ($\{B\}$, as seen in Chapter 2), can be obtained directly from a CAD model of a product design created in the CAD/CAM systems. However, difficulty frequently arises

in communication between the CAM systems and the NC-machine tools, especially when various machine tools are employed.

The interface that links the CAM systems and NC-machines is called the *postprocessor* and it converts CL-data to the specific controller input code needed. Essentially, different combinations of machine tool and control unit require different postprocessors. Consequently, a manufacturing system with a variety of machine tools requires several postprocessors.

Most CNC controller units are programmed using the widely established international standard ISO 6983 (G-code)¹ language, also known as RS274D, but it is not the unique existing code. In fact, many machine-tool manufacturers have introduced special features with the justification of improving or meeting new performances. Moreover, with the augmented scope of robotic manipulators, each brand has developed newer and non-standardized languages to command its motion.

4.4.1. Concept of *postprocessing*

A *postprocessor* is a *software link* in the CAD/CAM chain that communicates instructions from CAM to a CNC machine [7].

As a consequence, the *postprocessing* can be understood as the *adaptation* of the descriptive information of a machining process generated by a CAM system towards the specific numeric controller unit which drives the device used to machine a workpiece (Figure 4.9).

This descriptive information not only includes the CL-data (TCP position and orientation relative to the part frame), but also additional information such as tool changes or the occasional use of coolant systems. In the case of redundant manipulators, the use of the undetermined additional joints is expected.

As a conclusion, after the planning in a CAM system, a good postprocessor results in a code which fulfils a complete machining process, without need for editing and checking the code. On the contrary, it supposes a waste of time and entails risks which are non-desirable nowadays.

¹ New standards are being introduced in industry the later years for the newest machine-tools, for example ISO 10303 (also known as STEP) and ISO 14649 (STEP-CNC) [5]. Clearly both them are outside of the scope of this thesis due to the capabilities of current industrial robotic manipulators.

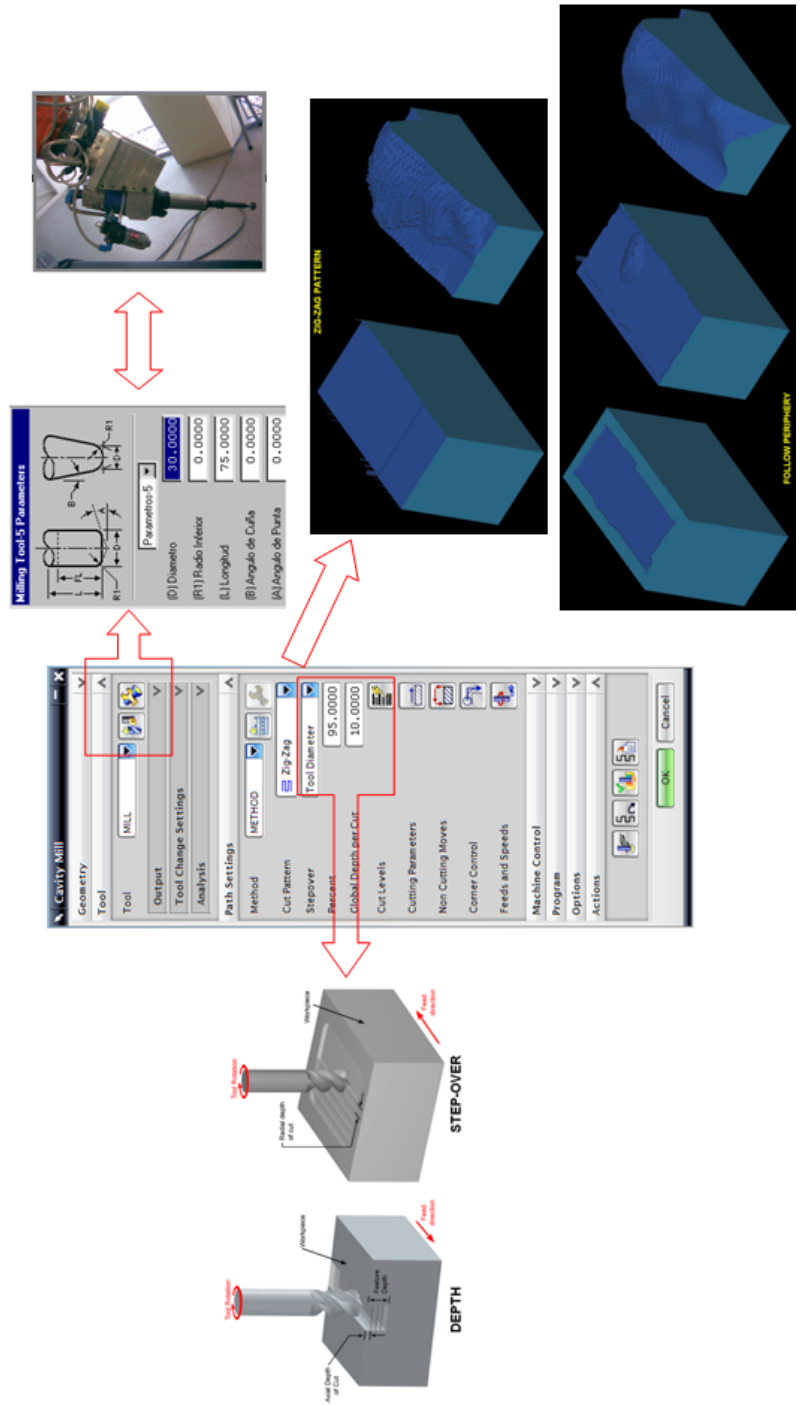


Figure 4.8. Input parameters prior to the generation of the toolpath (NX™)

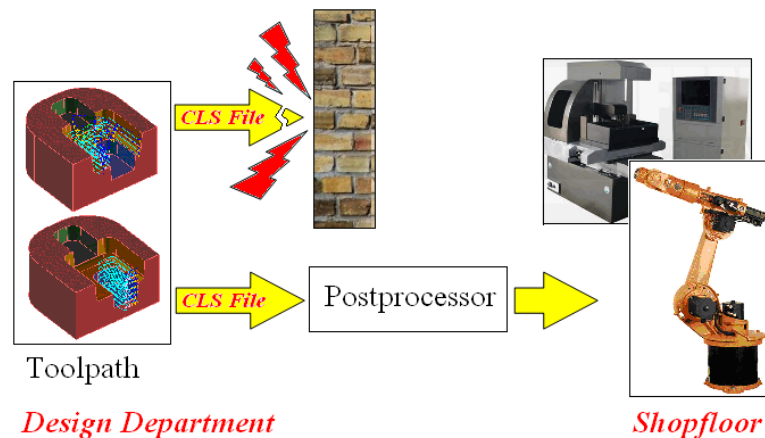


Figure 4.9. Concept of postprocessing as link between CAD/CAM and the production system at the shopfloor.

4.4.2. Literature review in CNC Postprocessing

The core research problems in NC appear to have been investigated in the 1970's and 1980's (i.e. path interpolation) and the emphasis of research activity has focused in applied NC in fields such as:

- Linking CNC machines in CIM environments
- Understanding of machining operations
- Tool life and efficiency diagnosis (material and management design).

For some authors, the first field (originally intended with the ISO-6983) has fallen behind due to the variety of machine configurations to which CNC has been applied. The required conversion or postprocessing is even more complicated because of the proliferation of CNC manufacturer-specific extensions [9].

It is also remarkable that CNC controlled milling machines and milling programming techniques have not featured strongly in the literature. It is surprising since milling represents one of the most demanding of all CNC applications. This could be due to the highly specialized nature of this class of production systems, but also this is probably due to the very expensive machinery required to test theories, placing this topic away from the reach of many research institutes.

A minimum of five axes are required in a milling machine to achieve the maximum possible position and orientation DOF's in the cutter relative to the

part frame {B} (see Chapter 2). The cutter tool, as rigid body located in Euclidean space, has three translational degrees of freedom and three orientation degrees of freedom but the third orientation degree of freedom is not required since the cutting tool possesses rotational symmetry about the spindle axis. This *functionally redundancy* inherent to the *pose* of the cutting tool will be discussed in Chapter 5. Now, from a practical point of view, it is remarkable that the calculation of the position of the tool centre point (TCP) and orientation of the tool axis is solved for all commercial software with five-axis capabilities (UnigraphicsTM, CatiaTM, PowermillTM, GibbsCamTM and others) [20]. Therefore, this matter is outside of all skilled CAM users' scope. The main trouble during toolpath generation appears in *postprocessing* steps, when the toolpath generated by the CAM system is translated into CNC code.

There are many different configurations of milling machines, and post-processors have to be adapted for each of them. A small numbers of researchers have recently worked on the basic architectural issues of CNC milling postprocessors. Bedi and Vickers [10] developed a postprocessor program for FANUC 6MB machine tool. In a similar way, Balaji [11] presented the development and implementation of a postprocessor by converting APT² source codes to a machine code format. However, both works are focused towards three-axis machines (i.e. static tool orientation). This fact makes the transformation from CL-data to NC-data straightforward and no additional coordinate transformation technique is necessary.

There are three typical five-axis machine tools proposed by Sakamoto and Inasaki [14], and different postprocessors have to be adapted for each of them. For example, using a machine with two rotary additional axes in bed (Figure 4.10, left) is totally different to those with two orientation angles (twist and tilt angles) in the tool head (Figure 4.10, right). Thus, the tool positions and orientations relative to the part frame require further transformations by the postprocessor before encoding it into the machine input language [20]. The transformation requires *knowledge of the kinematic architecture* of machine-tool in order to solve the inverse kinematics, i.e., transferring the tool positions and orientations (operation space) into machine axes positions (joint space). Lee and She [15] documented a postprocessor capable of converting cutter location CL-data to machine control data for those machines, in order to establish an interface between CAM-systems and those machines. For this, they made an inverse kinematic analysis of each sort to obtain the analytical equations for the resulting NC data. Recently, She et al. [16][17] have made a revision of this work.

² APT or Automatically Programmed Tool is a high-level computer programming language used to generate instructions for NC machine tools created by Douglas T. Ross during the late 1950's [19]. Today, it is an ISO standard of CL-file code [8][18].

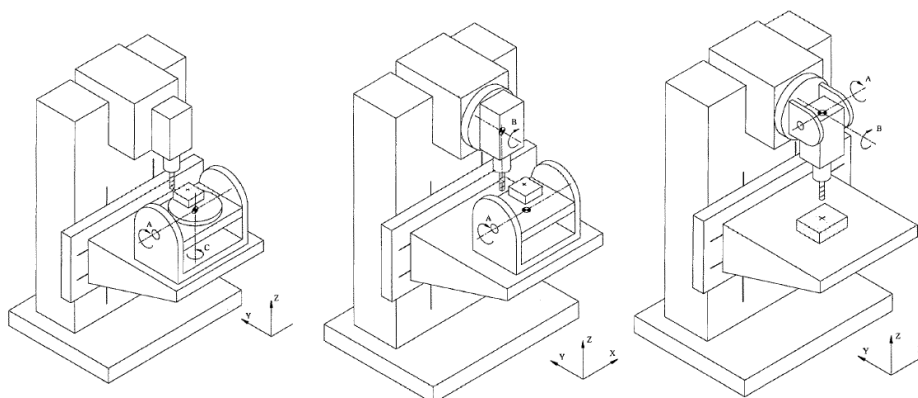


Figure 4.10. Three different configurations in a 5-axis machine tool [14].

Recent research have been done in some milling machines at the *University of Montreal* [21][22] to increase significantly their workspace by using its *functionally redundant* revolute joint, previously introduced. An optimization procedure was implemented within a postprocessor module and tested with the architecture of a generic five-axes milling machine [8]. They presented a *functional Redundancy-Resolution Scheme* (RRS) implemented within a postprocessor module of the generic B-Y-Z-X-C milling center Huron KX8-Five (Figure 4.11). The CL-data was generated with the commercial CATIA V5 system.

This previous research demonstrates that there is still tremendous scope for improvement in the basic *machine modelling* and *postprocessing* fields. Traditional CNCs are ill-suited to the demands of many of today's complex robotic workcells (including serial manipulators and parallel robots). Nowadays, these machines are characterized by intensive shop-floor level set-up and programming as opposed to the growing trend for CNC machines to be programmed off-line.

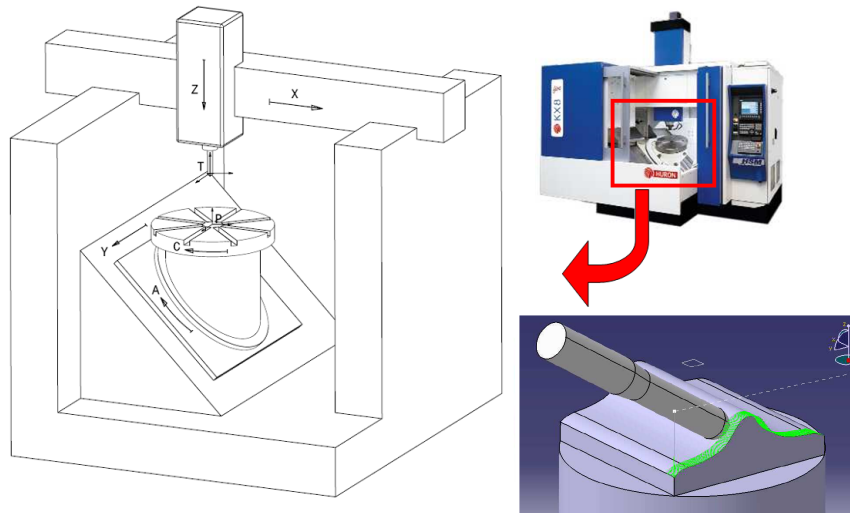


Figure 4.11. Generic B-Y-Z-X-C milling center Huron KX8-Five, and view of the CAM software CATIA.

Parallel Kinematic Machines (PKM) are clearly outside of the scope of this thesis, although it is remarkable the interest that they have achieved in a few years due to their major stability in fast milling machines, for example at [23][24]. For the author's point of view, it is worth mentioning the work recently done by Guo *et al.* [28] (Figure 4.12). These authors implemented a special postprocessor in TCL/TK language and based on UG/POST (NX-Siemens Corp.), and it was applied for converting the CL-data to machine control data.

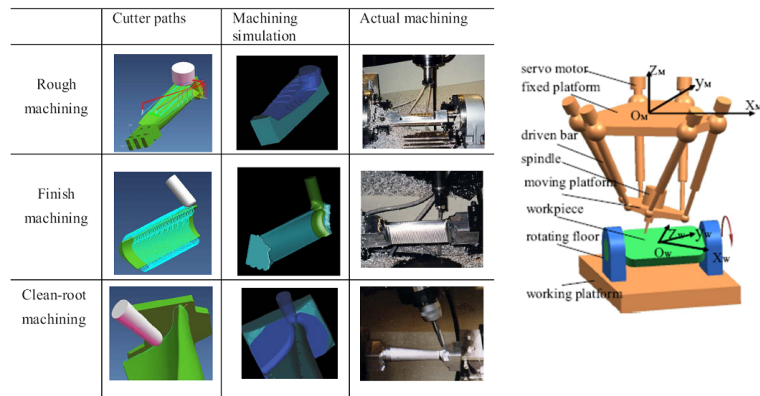


Figure 4.12. Guo *et al.* implemented a postprocessor for the NX system (Siemens Corp.) and to convert CL-data to PKM control data.

In case of serial robotic manipulators, as it is the scope of this thesis, the parameters and paths for the conventional CNC machining described above should be converted now into paths to be followed by a tool attached to the robot flange. Lorini and Meneghello [25] developed a computational application to translate CAD/CAM files into the programming language used in a 6R ABB™-robot (RAPID™). In a similar way, Feng-yun and Tian-sheng [27] developed a 6R robot system for complex surface polishing based on CL-data generated by a CAM system. For the author's point of view, it is worth mentioning the work recently done by Huang and Lin [26], since they developed a postprocessor to establish an interface between the UG (Unigraphics™) CAD/CAM system and the unique controller of a dual-robot workcell (Figure 4.13) by converting five-axis CL-data to robot control data. In order for both robots to perform cutting operations concurrently on the same workpiece, the original CL-data was divided into two parts, one for each robot in the workcell.

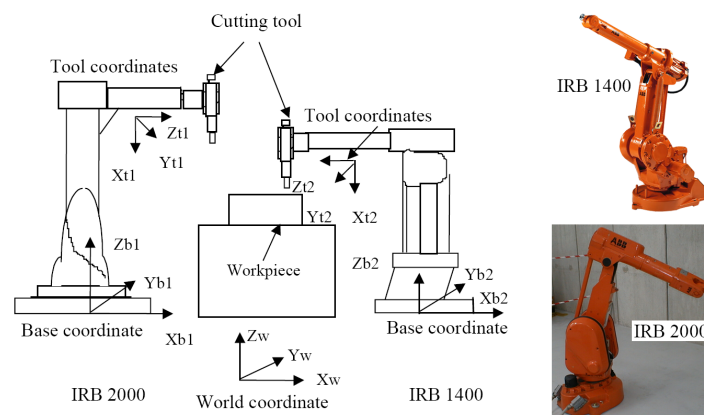


Figure 4.13. Huang and Lin converted five-axis CL-data into robot control data readable by the unique controller of a dual-robot ABB workcell

Finally, there is a discussion on the best way of programming postprocessors. The first way of developing postprocessors is based on the employment of sophisticated programming languages (Basic, Fortran, Pascal, C, C++, etc.) to develop the post-program in charge of the adaptation of the machine code according to several machine specifications (like the generalized NC postprocessor done by Ryu [29] by using Microsoft Visual C++) In general, this development requires a competent programmer. Whereas the method offers flexibility and potential to the developer, this way makes *extraordinarily difficult and costly* the creation and debugging of

postprocessors. Usually the last is a complex problem that could move in later responsibilities towards each particular user.

The second way rises from the programming languages specially developed for postprocessing. These are known as *interpreted languages* such as the previously introduced TCL/TK [37]. This has been the solution for the main CAM developers, being able to separate the tasks of *programming* and *debugging* postprocessors. These *interpreted languages* allow the development of hybrid interfaces that optimize a simple managing (within a range) of postprocessors for each type of machine-tool (Figure 4.14). As a result, CAM software developers leave the difficulties of a good postprocessor debugging as a labour for every particular user but usually sacrificing flexibility and programming potential by the restrictions of the interface. Some of CAM developers even provide collections of standard postprocessors to be adjusted by the user of a particular CNC machine.

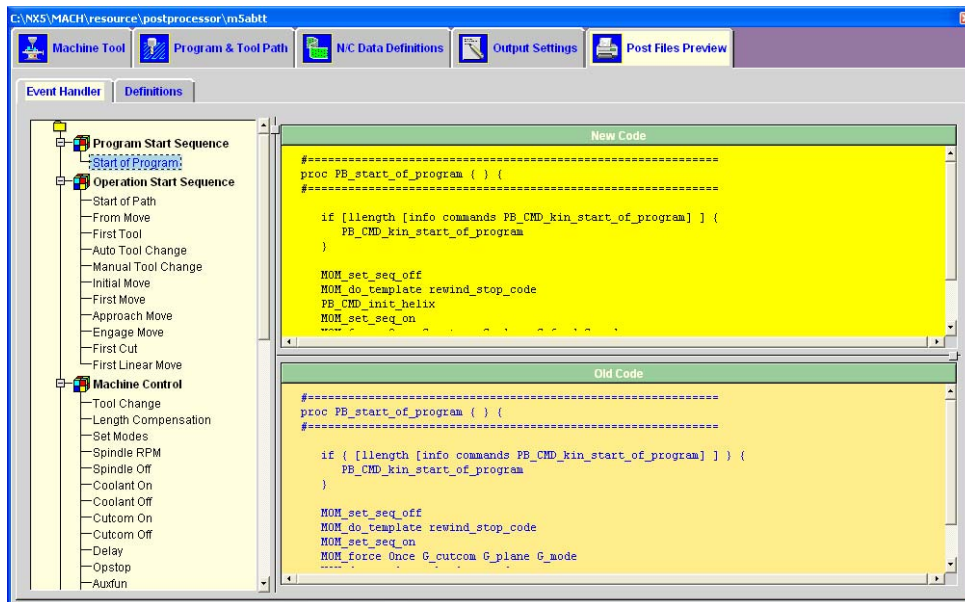


Figure 4.14. Post Builder interface that allows a simple managing of postprocessors for different standard machine-tools (within a range).

In case of complex mechanical systems like the robotic workcells shown previously (Huang and Lin; Guo *et al.*), or the one treated in this thesis, those commercial interfaces do not provide a way to implement the specific kinematic particularities.

4.4.3. CAM-ROB postprocessing

As introduced above, the NC programming code of a machine-tool can be generated directly for milling tasks up to 5-axes from any CAM system. The characteristics of these systems allow the automation of the process of NC programming through the introduction of operation parameters, manipulation of equipment or tool databases, and generation of the machining path (see Figure 4.8).

The CAM to ROB postprocessing considers the use of the above mentioned characteristics of the CAM systems for path generation (with the corresponding cutting parameters definition) and for the subsequent conversion of the toolpath to the specific robot language (KRL in the scope of this thesis). Thus, the postprocessor's objective is to interpret and manipulate the text file with the CL-data, in order to relate the different functions of the NC through the motion commands in robot language (hence, converting this information into a robot program).

As it will be shown in later sections, the application consists basically of programmed routines (such as TCL or C++ programming languages) with the purpose to automatically generate the robot language programming providing larger flexibility and automation to the operation of the robots, This allows diversifying their uses and reducing the processing time. The general structure of this application and its integration to the used systems is illustrated in Figure 4.15, where the referred application developed for the CAM-ROB postprocessing is marked within a border.

The graphical files generated in CAD systems are transferred to CAM systems by a graphical file transference standard format (iges, parasolid, step, ...) or more directly with the specific format of the CAD/CAM platform (*.prt in NX, as it is the program used in the IDF). The CAM software interprets the information from the CAD file and the toolpath to be is automatically generated according to selected parameters and strategies. The processes of circular interpolation or linear path generation are made by the algorithms implemented in the CAM module. The set of path information and operation parameters is recorded as CL-data.

From this point, the CAM-ROB postprocessor uses this CL-data or even the commonly generated CN program (usually G-code) as input data for the robotic system (see red border at Figure 4.15). The CAM-ROB postprocessor interprets this text file and correlates the functions in it with the specific functionalities and capabilities of the robotic system.

Finally, it is important to note that most of the current robotic CAD/CAM systems have powerful graphic capabilities that allow robot motion simulation. These approaches use the trial and error loop that is a time consuming

procedure applied by the technician, without use of any optimization concept, see Figure 5.12. It justifies the search of the optimum workpiece location with regard to the robot reference system {B} (see Figure 2.18) as done in Chapter 5, to automatically avoid *poor* manipulator postures (compared to other possible configurations).

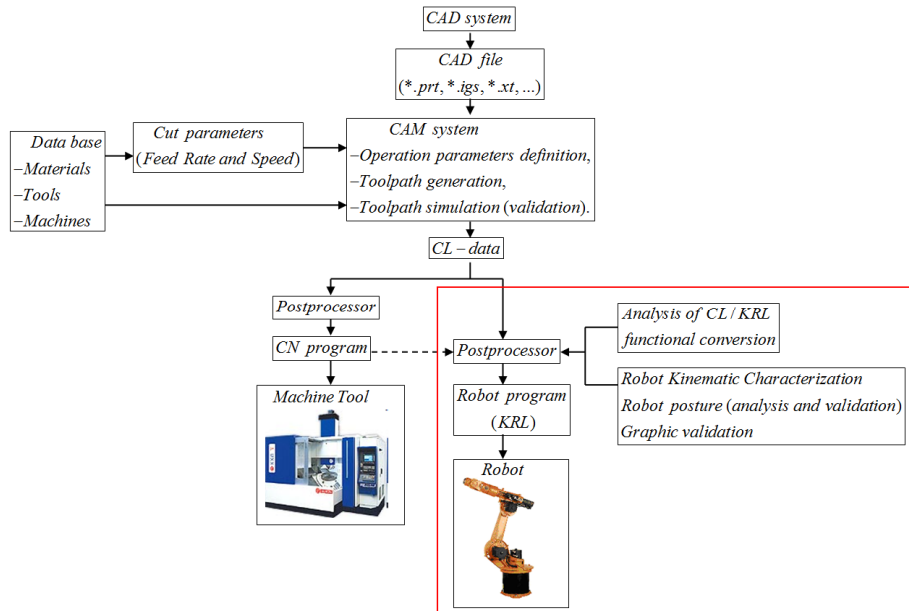


Figure 4.15. General structure of a CAM-ROB postprocessing

4.5. NX-CAM TOOLPATH GENERATION

4.5.1. NX-CAM module characteristics

NXTM (Siemens Corp.) is a digital product development system that integrates and fully associates the labours of design (CAD), simulation (CAE) and manufacturing (CAM). The *Computer Aided Manufacturing* (CAM) module makes possible the planning of milling tasks (Figure 4.2). In addition to the generation of the path planning of the successive tools (cutting strategies, speeds, etc), NXTM also allows to automatically reformulate the order of the successive machining operations.

i) Trajectory generation (CL-File). Linear and circular path tracking

As stated in Section 4.4.3. and even more with the advancing computer technology, commercial CAD/CAM systems can design complex

surfaces and directly generate their CL-data up to five-axis machining. In general, CAM systems work within three stages: the first is to generate *cutter contact points* on the CAD model, the second stage is to generate CL-data and the last stage is to convert CL-data to machine code.

Actually, with the parameters given by the designer (Figure 4.8) and to accomplish the two firsts stages, the CAM system automatically discretizes the path curve into small segments within the machining tolerance of the toolpath (Figure 4.16).



Figure 4.16. Trajectory tolerances, *intol* and *outol*, in the NX-CAM system.

Each of these end-points is saved in cutter location file (CL-file) under an ISO standard³. The five-axis CL-data consists of positions and orientations of the cutter with regard to the workpiece coordinate system {B}.

In order to execute the five-axis CL-data on the robotic workcell, the data are required to be transformed into different reference inputs. These reference inputs are normally *linear* motions or *circular* motions, although NURBS trajectories have been successfully implemented in the ultimate CN-machine tools and most CAM systems already are able to generate this approximation. For the scope of this thesis, only linear and circular motions are the possible CP trajectories most serial manipulators [26][33][34]. Thus, this would be the format data required from the NX-CAM (Figure 4.17).

³ One of these standards is the APT format (Automatically Programmed Tools), previously introduced.

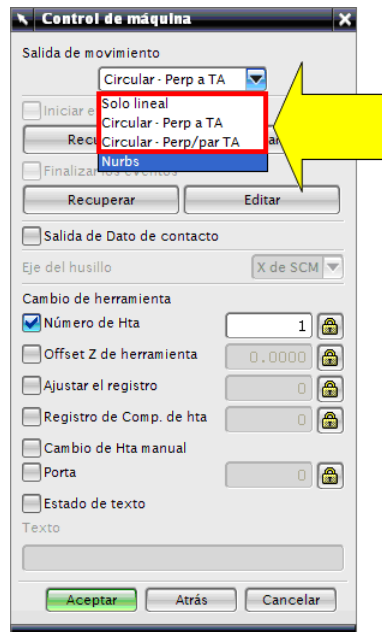


Figure 4.17. NX-CAM dialog window to determine the sort of CN inputs generated. Those inputs must describe *linear* and *circular* motions for most current industrial manipulators.

- *Linear and circular path tracking*

The most usual way to specify the toolpath in a milling operation consists of a succession of points with small linear interpolations between each two of them. Clearly, the approximation of a bended trajectory with small straight lines implies a loss of accuracy.

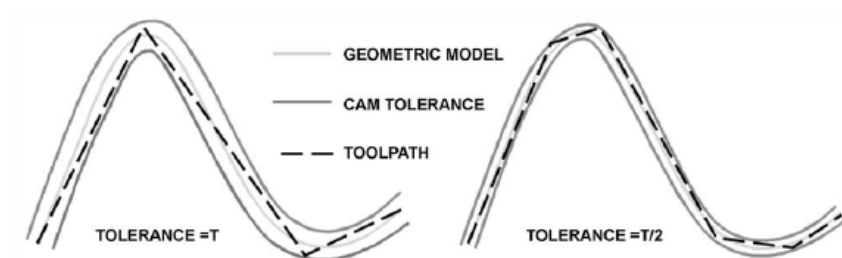


Figure 4.18. Influence of the tolerances (*intol* and *outol*) on the number of *linear* interpolations required to track a toolpath [38].

In case of trajectories with a small curvature radius, the density of points is higher than in case of nearly straight ones. The dialog box of Figure 4.16 is the way to control the degree of accuracy specified by the designer (*intol* and *outol* tolerances). Higher tolerances entail higher discretization error, resulting in a *faceted* piece. On the contrary, narrow tolerances force the CAM to make a great amount of interpolations resulting in a very long program (Figure 4.18). The influence of this discretization on the surface finish was recently documented by Helleno and Schützer [38], using the same CAM-system than in the current thesis.

An immediate solution to this problem consists of using arcs in each of the coordinate planes of the (XY-XZ-YZ) to approximate the toolpath (see Figure 4.19). It also reduces the size of the NC-program.

Although it is immediate to define the linear movement by the location of the destination point from the actual point, the definition of the circular movement depends on the controller unit. It varies with different criterion to define the arc center, the radius, the amplitude of the arc and the motion sense.

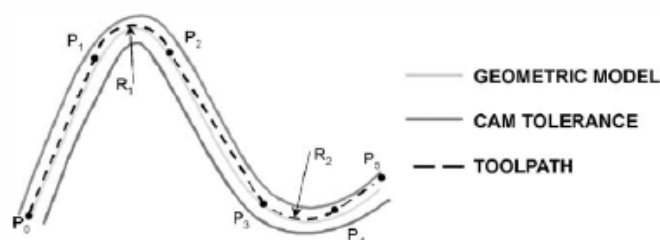


Figure 4.19. Toolpath interpolated with arcs and lines.

ii) NXTM-Post

As it was mentioned, the last stage of CAD/CAM systems is to convert CL-data to machine code, and the interface that links the CAM systems and NC-machines was already introduced as the postprocessor.

NXTM can also be considered as configurable postprocessing software [31][32], specifically named NX-Post. It uses the stored toolpath as input data, and provides a legible NC code. The Post postprocessor consists of several parts [40]: the *Event Generator*, the *Event Handler*, the *Definition File* and the *Output File*. The Event Generator is the NX core program that cycles through the events in a CL-file and communicates the data associated with each event to the Post postprocessor. An *event of path* is a collection of data, that when processed by Post, causes the NC machine

to perform some specific action [30][31]. For example, a basic *Linear_Move* event will cause the NC machine to move the tool along a straight line to a position specified by the information stored in the position parameters (X, Y and Z coordinates at {B}). In this case the *Event Generator* will trigger the *Linear_Move* event and will load the corresponding parameters X, Y, and Z with the values that represent the end position of the straight move postprocessed. A complete description of recognized events, and the variables associated with each, is described in [40] (at *Events* sub-section). By the way, the *Output File* is the file where the postprocessor writes the postprocessed instructions that will be read and executed by the NC-machine or robot.

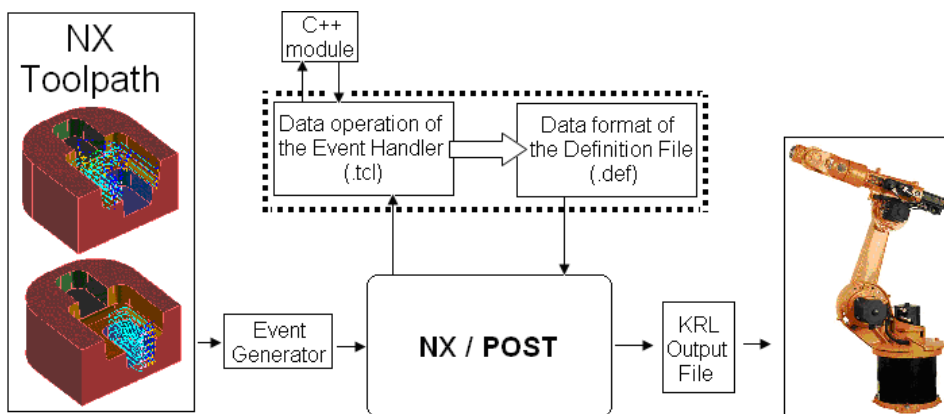


Figure 4.20. Integrated postprocessing in the NX-CAM system. The Definition File and the Event Handler are programmed in TCL to adapt NX's CAM towards the KUKA KRC2 controller. The Event Handler is able to interact with executable modules programmed in C++.

The great configurability of the NXTM integrated postprocessor is achieved by means of the interactivity of the system with two programs that manipulate the event variables and adapt the CAM data towards the particularities of the NC-machine (or the *robotic workcell*, Figure 4.20):

- *Event Handler* (.tcl): It is a file containing principally of a succession of procedures and calculations to carry out with the associated event variables to get the information required by the NC controller. For example, tool pose data relative to the part frame {B} may require further transformations before encoding it into the specific NC-file. For example, these transformations can be based on the machine tool architecture or the robot inverse kinematics transformations (as

studied by some authors, section 4.4.2.). For the scope of the IDF's workcell, special treatment is required by the KUKA™ KRC2 controller [33], see Section 4.6.3.

- *Definition File (.def)*: It is a static file that gives the required format to the output information, namely the desired toolpath. As example, the KRL structure [33][34] is the format required for the scope of this thesis, see Section 4.6. and 4.6.3.

Both programs are programmed in TCL (*Tool Command Language*), previously introduced in section 4.4.2. It is an interpreted programming language with capacity of connecting in a flexible way certain number of modules in other programming languages like C++ [35][36][37].

The Event Generator, the Event Handler, and the Definition File are dependent upon each other. Together they transform the tool path data contained in the part file into a set of formatted instructions that they can be read and executed by a specific machine tool/controller combination.

4.6. Industrial NX™ to KUKA™ workcell Postprocessing

4.6.1. KUKA™ Workcell programming

As the scope of this thesis is about using a KUKA™ robotic workcell for milling tasks, in addition of the architecture and kinematics of the workcell the input format required by the workcell controller must be known.

KUKA™ KRC2 controller uses textual language commands which are written in English-like statements to perform the motion program. The motion instructions can be subdivided into commands for simple PTP motions and commands for CP movements. Whereas, with CP, the EE describes a geometrically defined path in space (straight line or arc), the motion path in PTP movements is dependent on the robot's kinematic system and cannot, therefore, be accurately predicted in this industrial robot (see [34]). Common to both these types of motion is the fact that programming takes place from the current position to a new position. For this reason, a motion instruction generally only requires the specification of the end position (with the exception of CP circular motions)

Position coordinates can be specified either as text (writing the numeric values of the joints or tool coordinates) or by moving the robot to them and saving the actual values (by using an electronic keypad known as a *teach-pendant*). However, the last method cannot enter a program into the controller while the robot is off-line, and it is devoted to *pick & place* or *welding* programming.

Off-line programming has the advantage of preparing the robot program at a remote computer terminal, prior to it is deployed to the robot controller for its execution. Off-line programming systems generally include a computer graphic interface (a PC or laptop), which allows the robots to be programmed without access to the robot itself during the programming. It means the robot cell can be taught to perform a task via a computer while it continues to perform a separate task. Without off-line programming, workers must stop production, enter into the cell and manually walk the robot through the motions required to perform a task. Put succinctly, off-line programming has at least the following advantages [39]:

- reducing down-time caused by robot reprogramming;
- avoiding the risk of damage to real robot by checking the motion on a graphic simulator in advance;
- it becomes possible that existing CAD and CAM information are incorporate into the control functions.

Concerning these advantages, it is justified the attraction that the off-line programming has attracted a lot of robot developers (as shown in section 4.4.2.).

4.6.2. KRL for PTP motions (synchronous PTP)

In a KUKATM manipulator, the PTP motion is the quickest way of moving the TCP from the current position to a programmed end position [34]. To do this, the KRC2 controller calculates the necessary angle differences for each joint. The motions of the joints are synchronized in such a way that all of them start and stop moving at the same time (*synchronous PTP*).

The KRL (*KUKA Robot Language*) provides a specific command for PTP motions [34]:

$$PTP \{destination_point\} \quad (4.1)$$

The structure of the point required has two possible syntaxes:

$$\{E6POS: X, Y, Z, A, B, C, E1, E2\} \quad (4.2)$$

$$\{E6AXIS: A1, A2, A3, A4, A5, A6, E1, E2\} \quad (4.3)$$

In (4.2), the TCP positioning and orientation coordinates (X, Y, Z, A, B, C) are referred to the workpiece Cartesian coordinate system {B} placed on the rotary table (Figure 2.18).

In (4.3), $\{A1, \dots, A6\}$ set the desired final mechanical values for each joint of the main 6R manipulator. Those values correspond to the $\{\theta_1, \dots, \theta_6\}$ values calculated in the kinematic models of Chapter 2, respectively.

In both cases, $E1$ and $E2$ set the external joint values. In the case of the workcell studied in this thesis (Figure 2.14), they are the linear track d_L and rotary table θ_M joint values also calculated in Chapter 2, respectively. To practical effects, both additional external joint values are requested only in case of avoiding singularities or limit of range in the motion of the $A1$ - $A6$ chain (Figure 2.5). This criterion in the KRC2 controller attends to reasons of precision in the tasks (i.e., putting the calibration procedure on one side as treated in Chapter 3) and economy in the articulate motions. Also to avoid the mathematical problem that redundancy introduces, it is the criterion that KUKA™ takes, namely, leaving them aside and behaving as a 6R manipulator until the skilled technician perceives the convenience of its employment.

It is important to note that, as the mechanical origin or the rotation sense may show a discrepancy from the ones established in the DH modelling, some of them are offset by a constant. Moreover, the controller works with sexagesimal degrees ($^\circ$) instead radians (rad). Thus, readings from the controller (in $^\circ$) must be converted to rad and the values of the following table must be added to get the values in the DH kinematic model.

Mechanical joint	$E2$	$E1$	$A1$	$A2$	$A3$	$A4$	$A5$	$A6$
DH joint	θ_M	d_L	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Offset (rad)	π	0	π	0	$-\pi/2$	π	$-\pi$	π

Table 4.1. Offset between the mechanical and the DH modelled joint values

Finally, it is noteworthy that the precise path of a PTP motion between two points cannot be predicted exactly as the robot uses the quickest path it can. This path is influenced slightly by a number of factors [34]. This is the reason why PTP commands do not apply for controlling the actual industrial workcell for precise path tracking, as it cannot guarantee the precision and constant velocity required in the TCP for milling tasks.

4.6.3. KUKA KRL for Continuous Path Tracking

The KRL provides specific commands for *Continuous Path* tracking (CP commands). Unlike with PTP motions, in the case of CP motions it is not just

start and end positions what is predefined. Additionally, the accurate movement of the TCP along a path between these points is also mandatory. This path tracking may be a combination of *straight-linear* and *circular* TCP motions at a constant speed, commanded with the following templates [34]:

$$LIN \{destination_point\} \quad (4.4)$$

$$CIRC \{auxiliary_point\} \{destination_point\} CA \quad (4.5)$$

In both cases, the points required *must* follow the syntax of (4.2). The velocities to be entered do not relate any longer, to the individual axes, but to the motion of the TCP. The TCP is thereby moved at a precisely defined velocity.

In KRL, a circular movement has the peculiarity of that the sign of the circular angle (CA) does not indicate the rotation sense (like in the most common G-codes) but the order with which the TCP should move to the points. Figure 4.21 shows the effect of the CA sign in the CIRC command for the same points ($P_{AUX}=\{auxiliary_point\}$; $P_{PROGRAMED_END}=\{destination_point\}$) and amplitude (CA).

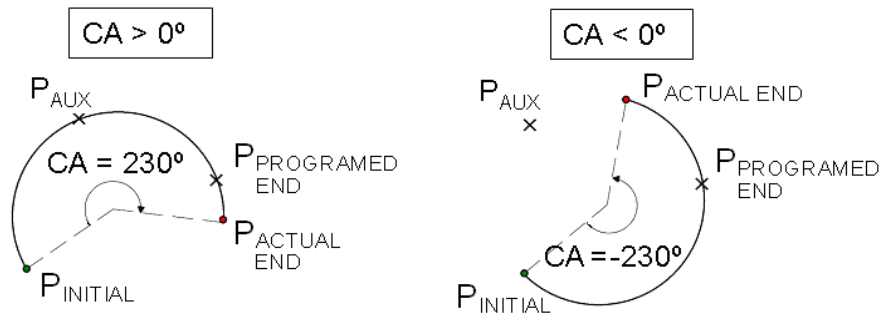


Figure 4.21. The actual end position ($P_{ACTUAL\ END}$) on the arc is determined by the programmed CA sign and value, and not by the destination point ($P_{PROGRAMED\ END}$)

Therefore, a circular movement is defined by three points different from one another and not on a straight line and a circular angle (CA) in degrees. Similarly, the variables that describe a circular movement in NXTM referred to a Cartesian Coordinate System {B} (Figure 2.23) are well known [40], and represented in Table 4.2. and Figure 4.22:

CONCEPT	VARIABLE	NAME OF THE EVENT HANDLER VARIABLE
Initial point	P_1	$[\$mom_prev_pos]_{3 \times 1}$
Arc center	P_C	$[\$mom_pos_arc_center]_{3 \times 1}$
Unitary vector, normal to the arc plane	N	$[\$mom_pos_arc_axis]_{3 \times 1}$ (left hand rule)
Arc angle	CA	$\$mom_arc_angle$ (degrees)

Table 4.2. NX Event Handler variables for circular movements.

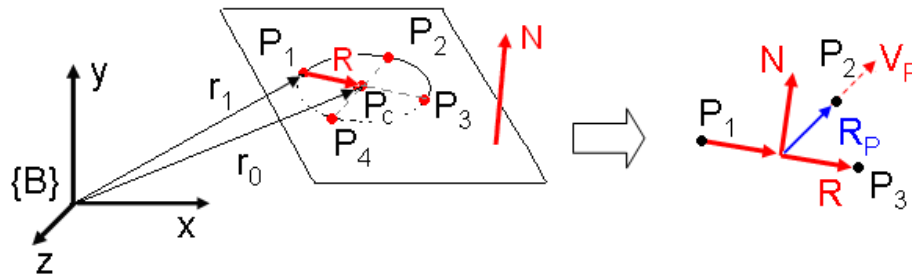


Figure 4.22. Left, Arc of circumference and arc plane. Right, definition points.

Thus, in order to obtain three points giving the equivalent circular motion at the manipulator, let r_i be the vector of position associated with P_i . Then:

$$R = r_0 - r_1 \quad (4.6)$$

$$r_3 = r_0 + R = 2 \cdot r_0 - r_1 \quad (4.7)$$

$$r_2 = r_0 + R_p \quad (4.8)$$

with

$$R_p = \frac{|R|}{|R \times N|} (R \times N) = \frac{|R|}{|V_p|} V_p \quad (4.9)$$

Those expressions allow getting the requested points. Additionally, the absolute value of the arc angle (CA) is well known, and always positive due to the way in which the points of destination and auxiliary have been defined.

4.6.4. Post programming

The information above, suitably treated by the *Event Handler*, is passed to the *Definition File*, which gives the required format for the output code ((4.4) and (4.5)). Appendix I shows the TCL code of both the *Event Handler* and the *Definition File* configuring a postprocessor from NX to a KUKA 6R manipulator, as the redundancy problem is treated in Chapter 5.

REFERENCES (Ch. 4)

- [1] Krouse, John K.; "What every engineer should know about computer-aided design and computer-aided manufacturing"; Marcel Dekker, Inc.; 1982, pp. 15-17, ISBN 0-8247-1666-3
- [2] Groover, M. P.; "Automation, production systems, and computer-integrated manufacturing"; Prentice-Hall, Inc.; pp. 699-711, 2001; New Jersey; ISBN 0-13-088978-4
- [3] Castillo García, F. J.; García Higuera, A.; CIM: El Computador en la Automatización de la Producción; Ediciones de la Universidad de Castilla-La Mancha, 2007. ISBN: 8484274446
- [4] McGraw-Hill Encyclopedia Of Science & Technology, McGraw-Hill, Editorial McGraw-Hill 2007, ISBN: 0071441433.
- [5] BODEMYR E., VALLIN D.; "How Improve a CAD/CAM/CNC-process, A study of organization and technology at Electrolux AE&T"; Department of Human Work Sciences, Luleå University of Technology, ISSN: 1402 – 1617 (2005)
- [6] R.-S. Lee and C.-H. She; Developing a Postprocessor for Three Types of Five-Axis Machine Tools; Int J Adv Manuf Technol (1997) 13:658-665
- [7] <http://www.toolingu.com/definition-300160-6999-post-processor.html> (accessed on 9th March, 2010)
- [8] Valipour, H. et Baron, L., The Orthogonal Decomposition Method in the Post-Processing of Redundant Machining Operations, Third International Conference on Industrial Automation, Montreal, Canada, (2007).
- [9] Gibbs B., "Postprocessors: An Integral Part Of Machine Tools"; Gibbs and Associates (www.productionmachining.com/articles/postprocessors-an-integral-part-of-machine-tools.aspx)
- [10] Bedi S. and Vickers G. W, "Postprocessor for numerically controlled machine tools", Computers in Industry, 9(1), pp. 3-18, 1987.
- [11] Balaji B. K., "Development and interface of a postprocessor for a CNC mill", Master Thesis, California State University, Long Beach, USA, December 1993.
- [12] Lin P. D. and Chu M. B., "Machine tool settings for manufacture of cams with flat-face followers", International Journal of Machine Tools and Manufacture, 34(8), pp. 1119-1 t 31, 1994.
- [13] Lin P. D. and Tsai I. J., "The machining and on-line measurement of spatial cams on four-axis machine tools", International Journal of Machine Tools and Manufacture, 36(1), pp. 89-101, 1996.
- [14] Sakamoto S. and Inasaki I., "Analysis of generating motion for five-axis machining centers", Transactions of the Japan Society of Mechanical Engineers, Series C, 59(561), pp. 1553-1559, 1993
- [15] Lee R.-S. and She C.-H., Developing a Postprocessor for Three Types of Five-Axis Machine Tools Int J Adv Manuf Technol (1997) 13:658-665, Springer-Verlag London Limited (1997)
- [16] She C.-H., Chang C.-C., Design of a generic five-axis postprocessor based on generalized kinematics model of machine tool, International Journal of Machine Tools & Manufacture 47 (2007) 537–545

- [17] She C.-H., Huang Z.-T., Postprocessor development of a five-axis machine tool with nutating head and table configuration, *Int J Adv Manuf Technol* (2008) 38:728–740, DOI 10.1007/s00170-007-1126-5
- [18] [http://en.wikipedia.org/wiki/APT_\(programming_language\)](http://en.wikipedia.org/wiki/APT_(programming_language)) (accessed on 19th March, 2010)
- [19] Ross, D.T.; Origins of the APT Language for Automatically Programmed Tools; *ACM SIGPLAN Notices*, Vol. 13, No. 8, August 1978
- [20] LOPEZ DE LACALLE L.N. , LAMIKIZ A., MUÑO A J. and SANCHEZ J.A.; The CAM as the centre of gravity of the five-axis high speed milling of complex parts; *International Journal of Production Research*, Vol. 43, No. 10, 15 May 2005, 1983–1999
- [21] Baron L., "An optimal surfacing post-processor module for 5-axes CNC milling machine", *Ecole Polytechnique de Montréal*, 2000.
- [22] Shareghi F. and Baron L., "An Optimal Redundancy-Resolution Scheme for the Post-Processing of 5-Axes Milling Machines", *Ecole Polytechnique de Montréal*, 2000.
- [23] Chen S.-L. and Liu Y.-C.; "Post-Processor Development for a Six Degrees-of-Freedom parallel-Link Machine Tool"; *Int J Adv Manuf Technol* (2001) 18:254–265
- [24] Chang T.-H., Chen S.-L., Liu Y.-C. and Inasaki I.; "Post-Processor Development of a Hybrid TRR-XY Parallel Kinematic Machine Tool"; *Int J Adv Manuf Technol* (2002) 20:259–269
- [25] Lorini F.J. and Meneghello G.P.; "A Milling System with Robot Resources"; *ABCM Symposium Series in Mechatronics - Vol. 1 - pp. 144-149* (2004)
- [26] Huang H.-k., Lin G.C.I.; "Rapid and flexible prototyping through a dual-robot workcell"; *Robotics and Computer Integrated Manufacturing* 19 (2003) 263–272
- [27] Feng-yun L. and Tian-sheng L.; "Development of a robot system for complex surfaces polishing based on CL data"; *Int J Adv Manuf Technol* (2005) 26: 1132–1137
- [28] Guo X., Wang L., Wang Z., Liu W.; "Development of CAD/CAM System for Parallel Kinematics Machine"; *Proceedings of the IEEE International Conference on Mechatronics and Automation*, pp. 2501-2507 (2007)
- [29] Ryu, G.-S., "IMPLEMENTATION OF WEB-BASED NC POSTPROCESSOR BUILDER"; *KSIAM IT series Vol.6, N°2*, 91-99 2002
- [30] A. González, J. C. Sánchez; "CAM UG, Manual cam completo v2.1."; UGS Corporation 2005
- [31] "NX Documentation"; ($\{\$UGII_base_dir\}$ UGDOC)
- [32] "Post Builder 5.0: Post Building Techniques"; UGS Corporation 2007
- [33] J. Andres, L. Gracia, J.Tornero; "Inverse kinematics of a redundant manipulator for CAM integration. An industrial perspective of implementation", unpublished (submitted to ICM09)
- [34] "KUKA System Software (KSS): Programación por el experto (KRC2 / KRC3)", Release 5.2., KUKA Corp., 2005.
- [35] B. B. Welch, K. Jones, J. Hobbs; "Practical programming in Tcl/Tk". ISBN 0130385603
- [36] F. Feito, R. J. Segura, F. de Asís; "Programación en Tcl/Tk", Universidad de Jaén, 1997. ISBN 8488942966

- [37] Ousterhout J. K.; "Tcl and the Tk Toolkit", Addison Wesley, 1994, ISBN 020163337X
- [38] Helleno A.L., Schutzer K.; "Investigation of toolpath interpolation on the manufacturing of die and molds with HSC technology"; Journal of Materials Processing Technology 179 (2006) 178–184
- [39] Celia F.; "Implant Manufacturers, Robotics Are Finding Common Ground"; <http://www.odtmag.com/articles/2005/11/implant-manufacturers-robotics-are-finding-common-.php> (Accessed 22nd March, 2010)
- [40] "NX Documentation - Post" ; (`{{ $UGII_base_dir }}\ UGDOC\ html_files\ ugpost\ index.html`)

CHAPTER 5

REDUNDANCY RESOLUTION SCHEMES

*“No scientist thinks
with formulae: before the
physician begins to calculate
he must have in his brain the
course of his reasonings.
Those, in most cases, can be
expressed with simple words.
Calculations and formulae
arise later” - A. Einstein*

CHAPTER 5. REDUNDANCY RESOLUTION SCHEMES

5.1. KINEMATIC REDUNDANCY

At Chapter 2, the operational space (Ω) was defined as the physical space where the manipulator changes the *pose* of its EE, with regard to a Base frame of reference $\{B\}$, in order to perform a task.

For a specific task, the motion of the EE may require the whole operational space Ω or only a subspace of Ω . In any case, the space in which the task is undergoing can be named as the *task space*, T , with $\dim(T) = t$.

In addition, m was presented as the dimension of the operational space¹; therefore, if the robot is required to position and orient its EE in three-dimensional Euclidean space, $m = 6$. Thus, for the final scope of this thesis, the following condition must be accomplished to perform a milling operation:

$$t \leq 6 \leq n, \quad T \subseteq \Omega \quad (5.1)$$

While most 6R manipulators have enough DOF's to perform a position and orientation tracking, it is known that their conditioning (and also its manipulability) may be frustrated due to mechanical limits or even worst, due to internal singularities (see Chapter 2).

5.1.1. Definition of Kinematic Redundancy

A manipulator is said to be redundant when the dimension of the task space t is less than the dimension of the joint space n [19], that is

$$n > t, \quad T \subseteq \Omega \quad (\text{with } n = \dim(\mathfrak{S}); t = \dim(T) \leq 6) \quad (5.2)$$

The degree of kinematic redundancy of the pair of serial manipulator-task, namely r_K , is computed as

$$r_K = n - t \quad (5.3)$$

¹ Although it could be considered $\dim(\Omega) = m \leq 6$, for the sake of this thesis only $m = 6$ will be taken into account. This is the case of general milling tasks, in which the milling tool can be positioned in all the x_i coordinates of (2.2).

It is also known that a redundant n -axis manipulator meant to perform tasks in the m -dimensional Cartesian-space has an $m \times n$ Jacobian matrix with $m < n$, as it was highlighted for the IDF's milling workcell in Chapter 2.

In contrast to non-redundant manipulators, it becomes now possible to avoid the above *frustrating* situations that could arise during a task. But at the same time, the control of a kinematically redundant manipulator is challenging since there are infinite possible joint trajectories for a given task.

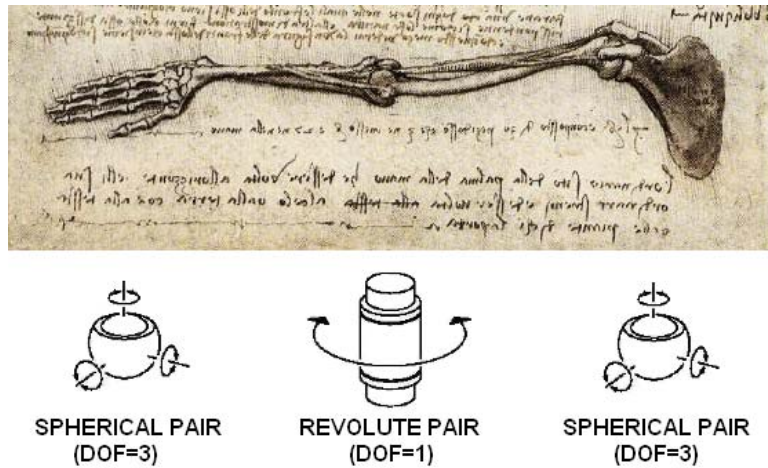


Figure 5.1. Anatomical studies of the arm showing the movements, by Leonardo Da Vinci (1510)

A typical example of a redundant manipulator is the human arm (Figure 5.1), which has 7 DOF from the shoulder to the wrist. If the shoulder and hand position and orientation are both fixed, requiring 6 DOF; the elbow can still be moved due to the additional mobility associate with the redundant DOF.

Previously introduced decoupled 6R manipulators (Figure 5.2) are widely used in industry because they are multipurpose. In fact, they can produce any position and orientation of the EE in their workspace. However, the concept of redundancy also can be related to the definition of the task instead as an intrinsic characteristic of the structure of the robot. Even if a manipulator is kinematically redundant for a specific task, it may not be redundant for another one.

For the scope of this thesis, devoted to milling tasks, it is interesting the visit the considerations that Huo and Baron [23][20] made on the concept of

redundancy. They distinguish two types of redundancy: *intrinsic* and *functional*. This classification, with different names is assumed by other authors [56].

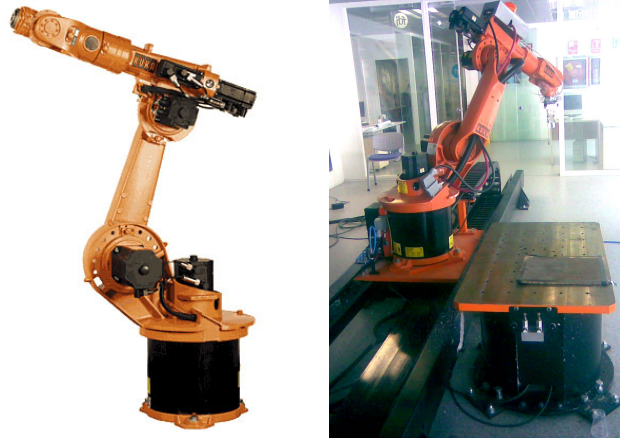


Figure 5.2. Left, decoupled 6R manipulator. Right, the same manipulator combined with two additional joints (linear track and rotary table)

i) Intrinsic redundancy

A serial manipulator for milling tasks is *intrinsically* redundant if

$$n > 6 \quad (\text{with } n = \dim(\mathfrak{S}); \dim(\Omega) = m = 6) \quad (5.4)$$

that is, the dimension of the *joint space*, \mathfrak{S} , is greater than the dimension of the resulting *operational space* of the EE, Ω , that is considered to be equal to six. The degree of intrinsic redundancy of a serial manipulator, namely r_i , is computed as

$$r_i = n - 6 \quad (5.5)$$

ii) Functional redundancy

The pair of serial milling manipulator and task are said to be *functionally* redundant when

$$6 > t \quad \text{with } T \subseteq \Omega, \quad t = \dim(T), \quad \dim(\Omega) = m = 6 \quad (5.6)$$

that is, the dimension of the *operational space*, Ω , is greater than the dimension of the *task space*, T , to be carried out by the milling tool attached to the EE. The milling task is also supposed to be totally included into the operation space of the manipulator, i.e., $T \subseteq \Omega$.

The degree of functional redundancy of a serial manipulator, namely r_F , is computed as

$$r_F = 6 - t \quad (5.7)$$

From (5.5) and (5.7) the kinematic redundancy (5.2) can be rewritten as

$$r_K = r_I + r_F \quad (5.8)$$

which makes clear that kinematic redundancy comes from two different sources: the *functional* redundancy and the *intrinsic* redundancy.

Milling tasks have a $t=5$, i.e. labours controlling position and two orientations of the tool (being these orientations constant or variable, namely *3-axes* or *5-axes milling*, respectively). In this case, there exists an axis around which a rotation of the EE is irrelevant, as shown in Figure 5.3.

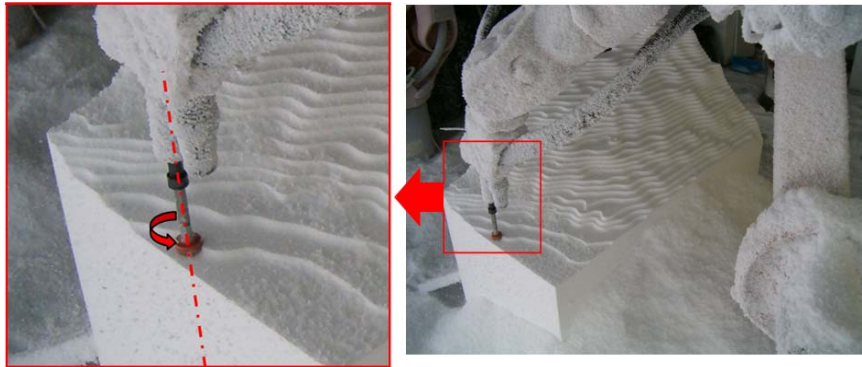


Figure 5.3. Irrelevant axis of symmetry of the tool at milling tasks.

If we reconsider the 6R manipulator of Figure 5.2 alone (left), it can be redundant when the task requires less than the full 6 DOF mobility of the EE, even more when additional joints are provided. For example, the *RP-6R* serial manipulator shown in Figure 5.2-right has a $\dim(\mathfrak{S}) = n = 8$ in an operational

space of $\dim(\Omega) = 6$. Hence, this manipulator has a degree of intrinsic redundancy of 2, i.e. $r_I = n - 6 = 8 - 6 = 2$. To define the *pose* of the milling tool, five reference inputs are required (i.e. three linear motions plus two rotational motions) and so $t = \dim(T) = 5$. Hence, the degree of functional redundancy is $r_F = 6 - t = 6 - 5 = 1$. As a conclusion, the kinematic redundancy of this pair manipulator-task is three: $r_K = r_I + r_F = 2 + 1 = 3$, as depicted in Figure 5.4.

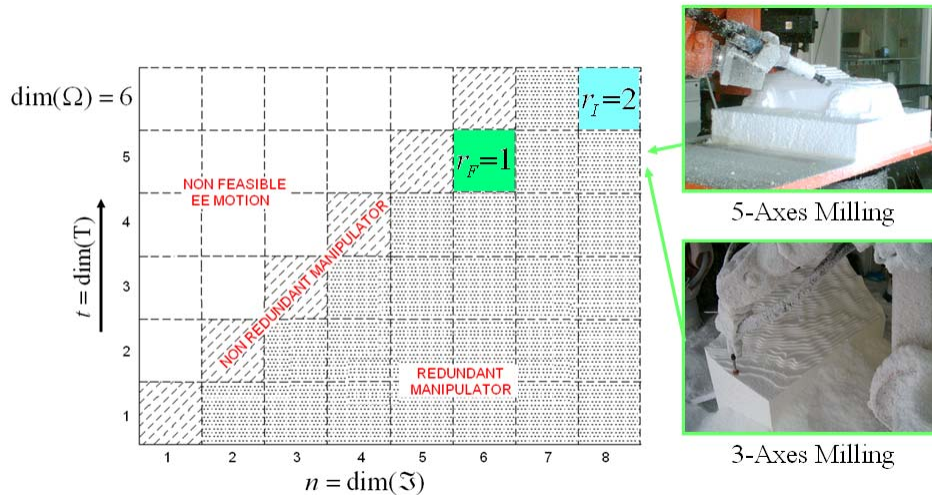


Figure 5.4. Intrinsic and functional redundancies of serial robotic tasks, with references to the studied workcell.

Summarizing, redundant manipulators have extra DOF's than those required to perform a main task. Solution strategies exploiting the potential benefits of these additional DOF's, are termed as *Redundancy Resolution Schemes* (RRS). To the author knowledge, it is notable that most of the reported RRS have been tested on simulations, while only a few implementations on *real* robots have been reported as Honegger and Codourey [24]. In many industries, the skilled operator is still who evaluates the best posture of the robot according to experience.

5.2. CONTINUOUS PATH PLANNING AND TRACKING

Once a continuous trajectory space is generated at Ω , the EE of the robot should track this trajectory, and hence, the joint angles of the robot have to be calculated along this continuous set of poses of the EE. In practice, the

continuous trajectory is sampled at a discrete set of close-enough poses $\{s_k\}_1^N$. A tangent, normal, and binormal unit vectors ($\{t, n, b\}$, respectively) can be associated with every sample point of the trajectory, namely the *Frenet-Serret* vectors, indicating the required pose (Figure 5.5).

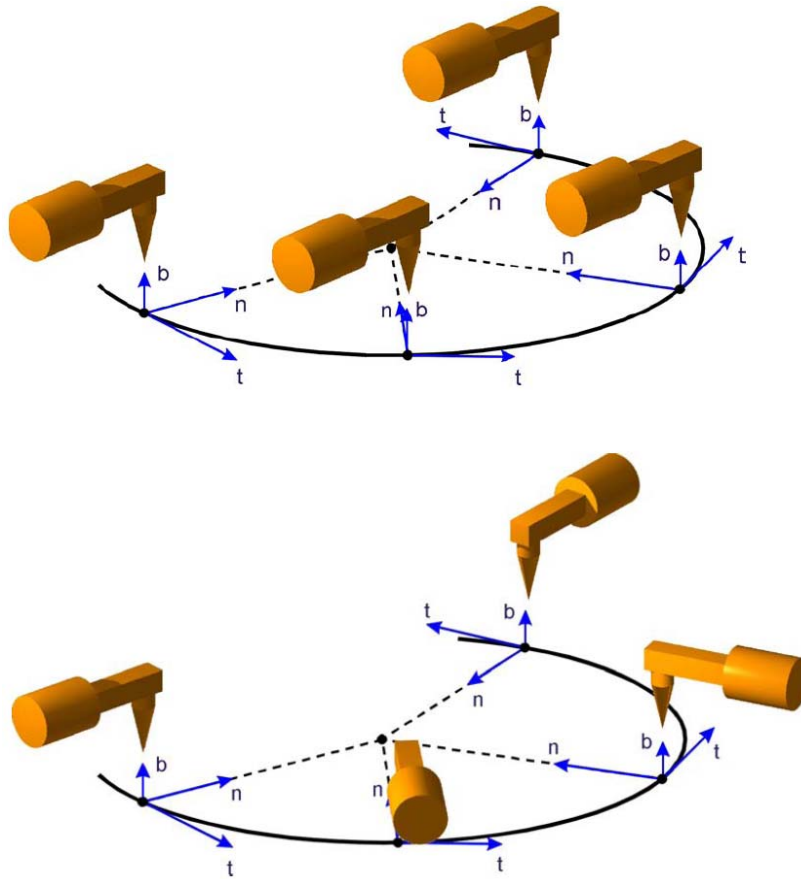


Figure 5.5. Milling toolpath with Frenet-Serret frames (*tangent t, normal n and binormal b*) indicating the required pose at each point in the toolpath. Again, it can be appreciated the irrelevant axis of symmetry of the milling tool.

In principle, an IKP of position could be solved at each sampled pose. If the manipulator is non-redundant and of the decoupled type these calculations are feasible in a fraction of a millisecond, as described in Chapter 2. However, if the manipulator has an architecture not lending itself to a simple solution, like the redundant workcell studied, an alternative approach is needed.

In order to deal with these inconveniences when solving the IKP, an iteratively approach can be applied as proposed by Angeles [48]. The procedure is based on Newton-Gauss method [47]. Recalling the DK statement, eq. (2.74), the desired toolpath at each pose can be expressed as a nonlinear algebraic system of the form

$$f(q) = s_k \quad (5.9)$$

Upon application of the Newton-Gauss method to find a solution of eq. (5.9) we assume an initial guess q^0 (usually a previous actual pose), and based on this value we generate a sequence q^0, \dots, q^i, q^{i+1} until either a convergence or an abortion criterion is met². This sequence is generated in the form

$$q^{i+1} = q^i + \Delta q^i \quad (5.10)$$

with Δq^i calculated from

$$J_a(q^i) \cdot \Delta q^i = s_k - f(q^i) \quad (5.11)$$

where J_a is defined as in (2.14).

Alternatively, the use of this differential form of the Jacobian matrix can be avoided as explained in Section 2.4.4., by using J_g [48][49]:

$$J_g(q^i) \Delta q^i = \Delta t^i \quad (5.12)$$

with Δt^i defined as

$$\Delta t^i \equiv \begin{bmatrix} Q_k \cdot \text{vect}(Q_k^T Q_d) \\ \Delta p \end{bmatrix} \quad (5.13)$$

where Q_k represents the actual rotation matrix from base frame to EE frame, Q_d represents the desired rotation matrix, and they have a relation as

$$Q_d = Q_k \cdot \Delta Q \rightarrow \Delta Q = Q_k^T \cdot Q_d \quad (5.14)$$

² It is common practice in all Newton methods to assume that a good enough approximation to the root wanted is available, and hence, $\Delta \theta$ is "small." Since any norm can be used to calculate the vector norm $\|\Delta \theta\|$, we can choose the norm that is fastest to compute, namely, the *Chebyshev norm* (this norm only requires comparisons and no floating-point operations): $\|\Delta \theta\|_\infty \equiv \max_i \{|\theta_i|\}$

Function $\text{vect}(\Delta Q)$ represents the axial vector of a 3x3 rotation matrix ΔQ , and is calculated as [48]

$$\text{vect}(\Delta Q) \equiv \frac{1}{2} \begin{bmatrix} \Delta Q_{32} - \Delta Q_{23} \\ \Delta Q_{13} - \Delta Q_{31} \\ \Delta Q_{21} - \Delta Q_{12} \end{bmatrix} \quad (5.15)$$

Vector Δp is defined as the difference between the prescribed value p_d of the position vector of the operation point and its actual value p_k .

The relations amongst Q_d, p_d, Q_k, p_k , and $\Delta Q, \Delta p$ are shown in Figure 5.6.

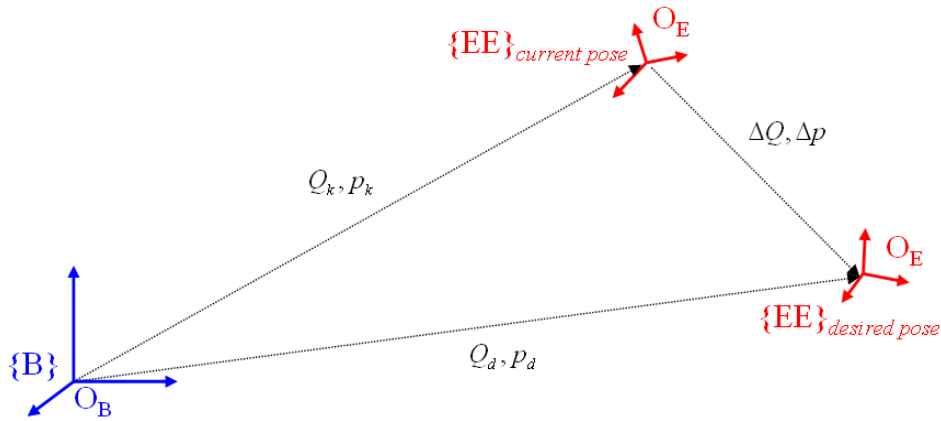


Figure 5.6. Relation between the *desired* and the *current* pose.

From Angeles [48], Baron and Huo [20][22][23] sketched an algorithm applied it to their RRS (exposed in the following section). For the scope of this thesis, a generic algorithm (5.16) can be deduced, from which some variations in the 8th step will be done in following sections:

- 1) $q \leftarrow$ initial joint position
- 2) $\{p_d, Q_d\} \leftarrow$ desired EE pose
- 3) $\{p, Q\} \leftarrow$ DK(q)
- 4) $\Delta Q \leftarrow Q^T \cdot Q_d$
- 5) $\Delta p \leftarrow p_d - p$
- 6) $\Delta t \leftarrow \begin{bmatrix} Q \cdot \text{vect}(\Delta Q) \\ \Delta p \end{bmatrix}$ (5.16)
- 7) $J_g(q) \leftarrow$ DK(q)
- 8) $\Delta q \leftarrow$ Redundancy Resolution Scheme
- 9) if $\|\Delta q\| \leq \varepsilon \Rightarrow$ STOP, else
- 10) $q \leftarrow q + \Delta q$, and go to step 3

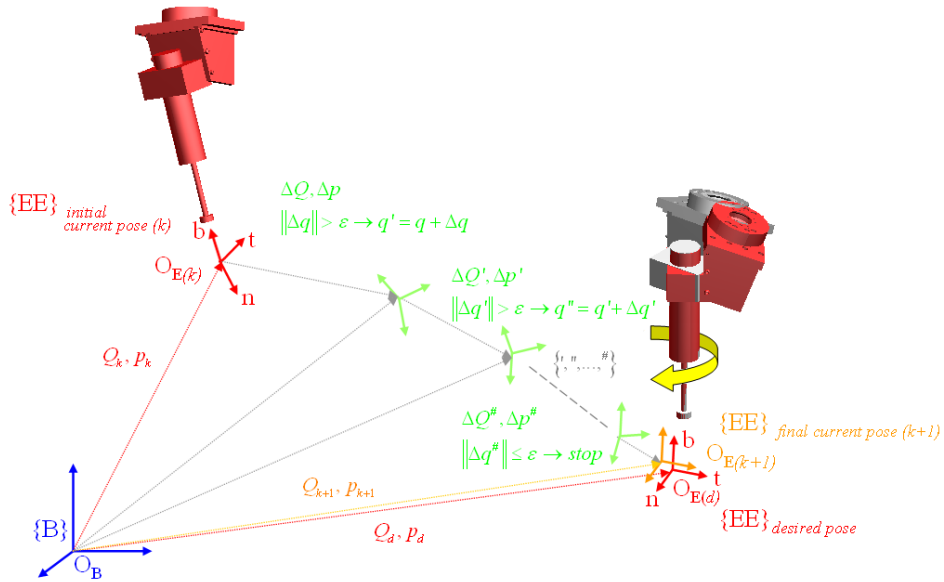


Figure 5.7. Highlight of the loop leading from an initial current pose (k) to a desired final pose.

5.3. REDUNDANCY RESOLUTION SCHEMES (RRS)

Depending on the application requirements and choice of controller, redundancy can be resolved at joint-position [13], velocity [15], or acceleration level [19]. In practical terms, at most robot controllers the control input is written

in form of a joint-position or EE-position values, but due to the fact that this problem is highly non-linear, a proper analysis of position can be done from the RRS at joint-rate level.

At Figure 5.8, the vector $\dot{q} \in R^n$ is mapped into $t \in R^m$ ($m = 6$). Two fundamental subspaces associated with a linear transformation are its Null Space ($\aleph(J)$) [3] and its Range ($\aleph(J)$), namely

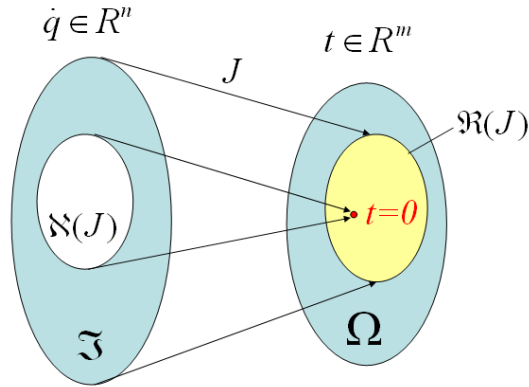


Figure 5.8. Mapping at velocity level, from the Joint Space (\mathfrak{S}) to the accessible workspace ($\aleph(J)$) in the Operational Space (Ω).

$$\aleph(J) = \{ \dot{q} \in R^n \mid J \cdot \dot{q} = \mathbf{0}_{n \times 1} \} \quad (5.17)$$

$$\aleph(J) = \{ J \cdot \dot{q} \mid \dot{q} \in R^n \} \quad (5.18)$$

From the fact that J is a linear transformation of R^n , Equation (5.17) means that $\aleph(J)$ is the set of all vectors \dot{q} such that are mapped to the null vector, $\mathbf{0}_{n \times 1}$. On the other side, only a sub-space of the Operational Space will be reachable, being it $\aleph(J)$.

From a mathematical point of view, the SVD³ of an $m \times n$ Jacobian matrix J can be written in the form

$$J = U \Sigma V^T \quad (5.19)$$

where U is the $m \times m$ orthonormal matrix of the output singular vectors $\{u_i\}$, V is the $n \times n$ orthonormal matrix of the input singular vectors $\{v_i\}$, and $\Sigma = [S \ 0]$ is the $m \times n$ matrix whose $m \times m$ diagonal submatrix S contains the singular values σ_i of the matrix J (strictly positive). Letting $\text{rank}(J) = r$, the following holds:

³ <http://mathworld.wolfram.com/SingularValueDecomposition.html>

- $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_n = 0$
- $\mathfrak{R}(J) = \text{span}\{u_1, u_2, \dots, u_r\}$
- $\mathfrak{N}(J) = \text{span}\{v_{r+1}, v_{r+2}, \dots, v_n\}$

From a practical point of view, the joint velocities belonging to $\mathfrak{N}(J)$, can be specified without affecting the task space velocities, since they do not affect the motion of the EE. They are referred to as internal joint motion or *self-motions*, and there are *infinite options*. This shows the major advantage of redundant manipulators: additional objectives can be satisfied while executing a main task specified via positions and orientations of the EE, such as occurs in milling tasks.

Several authors [17][19] classify the RRS into two trends, namely *local* and *global* methods:

- The *global methods* need all the required data before the movement is realized in a time invariant workspace, for tasks requiring strict optimality. Those methods, involve a great amount of computations making prohibitive a fast resolution scheme [17][19].
- The *local methods* search a solution for every instant with the use of the available data. Those schemes would be found the more convenient for the treatment of a CAM-Rob postprocessing task, for two reasons: the immense amount of data and the major flexibility in the treatment of the process.

5.3.1. Local Optimization Algorithms for intrinsically-redundant manipulators (r)

Most of the RRS focus on the solution of intrinsically-redundant manipulators, by using $\mathfrak{N}(J)$ to select an optimized solution, and use the Moore-Penrose pseudo inverse (or a weighted pseudo inverse) of the Jacobian matrix.

i) Schemes with the Moore-Penrose Pseudo-Inverse

For a redundant manipulator, the dimension of $\mathfrak{N}(J)$ is equal to $n-r$, where r is the rank of the matrix J . If J has full column rank ($r=m$), then the dimension of $\mathfrak{N}(J)$ is equal to the degree of redundancy [33].

For these RRS, we recall the eq. (2.13), namely

$$\dot{q} = \overbrace{J^\dagger t}^{\text{Minimum-norm solution}} + \overbrace{(I - J^\dagger J)h}^{\text{Homogeneous solution}} \quad (5.20)$$

Equation (5.20) has been used by many researchers in order to solve redundant tasks (see [1][2][5][9][25]). The ability of the pseudo-inverse to provide a meaningful solution in the least-squares sense regardless of whether equation $t = J \cdot \dot{q}$ is underspecified, square, or over-specified makes it the most attractive technique in redundancy resolution.

The first part of (5.20) is the minimum-norm solution⁴ or *base solution* and the second part is an arbitrary vector from the Null Space⁵ of the Jacobian, being $(I - J^\dagger J)$ the *projection operator* on $\mathfrak{N}(J)$.

Vector h of (5.20) is an *optimized performance criterion vector* (*performance vector* for shake of brevity). Namely, the manipulator is required to track a desired target positions as *primary task*, but in addition one can try to accomplish secondary goals by properly choosing h . In this case, the *performance vector* can be taken as a *virtual force* which attempts to push the configuration of the manipulator away from the critical area of configuration space [38].

- **Selection of the performance vector, h**

Different selections of h result in different performance methods, most related to various applications of RRS.

The most widespread method used to apply such a secondary motion criterion through the Null Space (the primary requirement being a prescribed end point motion in the workspace) is the *Gradient projection method* (GPM), introduced by Liégeois [4]. It takes the minimization a

⁴ If t is in the range of J , $t = J\dot{q}$, \dot{q} is the unique vector solution of smallest magnitude. If t is not in the range of J , \dot{q} is the unique vector of smallest magnitude which minimizes $\|J\dot{q} - t\|$, or equivalently, which minimizes $\|J\dot{q} - t\|^2$. [29]

⁵ Clearly: $J\dot{q} = J(J^\dagger t + (I - J^\dagger J)h)$
 $J\dot{q} = JJ^\dagger t + J(I - J^\dagger J)h$
 $J\dot{q} = \underbrace{JJ^\dagger t}_t + \underbrace{(J - (JJ^\dagger)J)h}_0 = t$

position-dependent scalar performance criterion (or virtual potential function [38], $p(q)$), by means of its gradient vector (h), namely

$$h = -k \cdot \nabla p \quad (5.21)$$

$$\nabla p = \left[\frac{\partial p(q)}{\partial q_1}, \frac{\partial p(q)}{\partial q_2}, \dots, \frac{\partial p(q)}{\partial q_n} \right]^T \quad (5.22)$$

With the aim of avoiding joint limits, Liégeois [4] introduced a performance criterion that helps joint-limit avoidance with the lower and upper joint limits known (q_i^{\min} and q_i^{\max} , respectively)

$$p = \frac{1}{n} \sum_{i=1}^n \left(\frac{q_i - q_i^{\text{mid}}}{q_i^{\text{mid}} - q_i^{\text{max}}} \right)^2, \quad \text{with } q_i^{\text{mid}} = \frac{q_i^{\min} + q_i^{\max}}{2} \quad (5.23)$$

A variation of this criterion has been recently applied by Huo et al. [23].

With the aim of avoiding singularities, Yoshikawa [5] suggested the measure of manipulability (μ), introduced in Chapter 2, as performance criterion (i.e., $p \equiv \mu(q)$). Several authors have used the manipulability as the distance criterion to stay away from manipulator singularities [36]. He also introduced a performance index for obstacle avoidance, namely

$$p = \frac{1}{2} (q - q_r)^T W (q - q_r) \quad (5.24)$$

where W is a diagonal weighting matrix and q_r is a given arm reference posture.

Also the condition number of J , has been referred as singularity avoidance criterion [23] but, to the author's knowledge, by using the matrix 2-norm (i.e., the ratio of the maximum and minimum singular values of J termed as $k_2(J)$). No direct application of the condition number derived from weighted Frobenius norm (see Chapter 2) has been reported. Huo and Baron [23] combined the manipulability and the k_2 -condition number in a single index that they named as parameter of singularity (ω_{ps})

$$\omega_{ps} = \sqrt{\frac{k_2}{\mu}} = \sqrt{\frac{\sigma_1 / \sigma_m}{\sigma_1 \cdot \sigma_2 \cdot \dots \cdot \sigma_m}} = \sqrt{\frac{1}{\sigma_2 \cdot \dots \cdot \sigma_m^2}} \quad (5.25)$$

It is logical to activate the *performance criterion* related to singularity avoidance when the *parameter of singularity* considered passes over a preset *threshold* value q_{Ts} . At that instant, the corresponding configuration q is recorded. Thus, the parameter of singularity can be written as:

$$p_{ps} = \frac{\omega_{ps}}{2} (q - q_{Ts})^T W (q - q_{Ts}) \quad (5.26)$$

In (5.24) and (5.26) the choice of the *weight*, W , is a major difficulty to implement due to the subjectivity for a given *performance criterion*, p . At the same time, it is critical for the performance of the RRS. In particular, a small value of the W may slow down the minimization of the performance criteria, but on the other hand a large value may even lead to an increase of p [33]. Therefore, W is usually set based on trial and error [23].

In practice, it seems to be desirable to identify an appropriate value of W at each configuration in a reasonable time.

ii) Schemes using the Weighted Pseudo-Inverse

Several authors [37] propose that rather than driving the robot away from singularities at very high demands in joint velocities, the GPM solution based on the $\aleph(J)$ projection sometimes leads the robot to singularities. A weighted pseudo-inverse by the inertia matrix can be used instead. Thus, a weighted solution is directly deduced from (5.20), namely

$$\dot{q} = J_w^\dagger t + (I - J_w^\dagger J) \cdot h \quad (5.27)$$

where a weighted pseudo-inverse (WPI) is used

$$J_w^\dagger = W^{-1} J^T (J W^{-1} J^T)^{-1} \quad (5.28)$$

and W is a positive-definite weighting matrix.

Recently, Honnegger [24] implemented this solution for the control of the redundant manipulator Robojet®.

iii) Schemes using Householder Reflection

Arenson, Angeles and Slutski [1][18] proposed to use Householder reflection in a RRS. At first, equation (5.20) can be rewritten as:

$$\dot{q} = k + h \quad (5.29)$$

with

$$k = J^\dagger(t - Jh) \quad (5.30)$$

Multiplying both members of (5.29) by J , then

$$J\dot{q} = Jk + Jh \quad (5.31)$$

Therefore,

$$Jk = r \quad (5.32)$$

where $r = t - Jh$.

For solving (5.32), Householder reflection is used for the transposed Jacobian matrix J^T . The matrix H and U are reached, and they have a relation with J^T in the form

$$H_{n \times n} J^T = \begin{bmatrix} U_{m \times m} \\ \mathbf{0}_{(n-m) \times m} \end{bmatrix} \quad (5.33)$$

where U is a $m \times m$ upper-triangular matrix, H is an orthogonal matrix⁶ ($n \times n$), and $n > m$ for redundancy⁷. Hence, from (5.32) there is

$$JH^T Hk = (HJ^T)^T Hk = r \quad (5.34)$$

⁶ $H^T H = H H^T = I$

⁷ in this research, $m=6$

As

$$HJ^T = \begin{bmatrix} U \\ 0 \end{bmatrix} \Rightarrow (HJ^T)^T = [U^T \ 0^T] \quad (5.35)$$

equation (5.34) is equal to

$$\begin{bmatrix} U^T & 0^T \end{bmatrix} Hk = r \quad (5.36)$$

If we define $Hk \equiv y = [y_1^T \ y_2^T]^T$, where y_1 is an m -dimensional vector and y_2 is an $(n-m)$ -dimensional vector, then Arenson notes that k and y have the same Euclidean norms. Hence, minimizing $\|k\|_2$ is equivalent to minimizing $\|y\|_2$. Then, if we want to minimize $\|k\|_2$, or equivalently, $\|y\|_2$, we can choose $y_2 = 0$ and y_1 will be found from (5.36)

$$\begin{bmatrix} U^T & 0^T \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = r \xrightarrow{y_2=0} U^T \cdot y_1 = r \quad (5.37)$$

Now, with y known, k can be found from the previous definition of y with the fact that H is an orthogonal matrix

$$k = H^T \cdot y \quad (5.38)$$

which we can substitute in (5.29)

$$\dot{q} = H^T \cdot y + h \quad (5.39)$$

We should not forget that these computations will be performed with finite precision, and hence, roundoff-error amplification is bound to occur. In order to keep roundoff-errors as low as possible, this algorithm avoids the direct calculation of the generalized inverse of the Jacobian matrix, as with (5.20). Hence, the squaring of the condition number of the J is avoided and the round-off error of the algorithm is not amplified.

It is an interesting solution for the calculation of ill-conditioned postures because here the Jacobian matrix J may have a very high condition number.

iv) Schemes using the damped least-squares (DLS-) inverse

In the field of RRS, a problem with (5.20) is its instability around a singularity: in some circumstances, postprocessing a toolpath tracking with (5.20) seems to be mathematically cumbersome. More precisely, the norm of the first term of (5.20) becomes very large in the immediate neighbourhood of these configurations. Thus, there are some velocities in task space which require physically unrealizable joint rates. To deal with this inconvenience, Wampler [30], and Nakamura and Hanafusa [26], introduced the *DLS-method*. In essence, it minimizes

$$\|J \cdot \dot{q} - t\|^2 + \lambda^2 \|\dot{q}\|^2 \quad (5.40)$$

where $\lambda \in R$, the *damping factor*, is used to specify the relative importance of the norms of joint rates and the tracking accuracy. This is equivalent to minimizing the quantity of a new augmented system of equations [29], namely

$$\left\| \begin{bmatrix} J \\ \lambda I \end{bmatrix} \dot{q} - \begin{bmatrix} t \\ 0 \end{bmatrix} \right\| \quad (5.41)$$

The corresponding normal equation is

$$\begin{bmatrix} J \\ \lambda I \end{bmatrix}^T \begin{bmatrix} J \\ \lambda I \end{bmatrix} \dot{q} = \begin{bmatrix} J \\ \lambda I \end{bmatrix}^T \begin{bmatrix} t \\ 0 \end{bmatrix} \quad (5.42)$$

This can be equivalently rewritten as

$$(J^T J + \lambda^2 I) \dot{q} = J^T t \quad (5.43)$$

which leads to

$$\dot{q} = J_a^{\dagger \lambda} t \quad (5.44)$$

with

$$J_a^{\dagger \lambda} \equiv (J^T J + \lambda^2 I)^{-1} J^T \quad (5.45)$$

It is easy to show [29] that

$$(J^T J + \lambda^2 I)^{-1} J^T = J^T (J J^T + \lambda^2 I)^{-1} \quad (5.46)$$

with

$$J_b^{\dagger \lambda} \equiv J^T (J J^T + \lambda^2 I)^{-1} \quad (5.47)$$

The advantage of $J_b^{\dagger\lambda}$ over $J_a^{\dagger\lambda}$ is that the matrix being inverted is $m \times m$ instead $n \times n$, and m is less than n in redundant manipulators. Thus, we can rewrite (5.44) as

$$\dot{q} = J_b^{\dagger\lambda} \cdot t \quad (5.48)$$

It is important to mention that it is a frequent mistake to use $J^{\dagger\lambda}$ for the construction of the homogeneous term of (5.20), as the damped least squares inverse lacks various vital properties [27]. Instead, it must be built as done in (5.20), i.e., based on J^\dagger .

The singular value decomposition (SVD), introduced at the beginning of this Section, provides a powerful method for analyzing the *Pseudo-Inverse* and the DLS-methods [26]. From the previous section (i), let's consider J not having full column rank (i.e., $r < m$). With this consideration, the last $r-m$ singular values of Σ (5.19) are zero, $\mathfrak{R}(J)$ is an r -dimensional subspace of R^m , and the dimension of $\mathfrak{N}(J)$ increases to $n-r$. Recalling (5.19), the SVD of an $m \times n$ Jacobian matrix J of rank r can be written in the form:

$$J = \sum_{i=1}^r \sigma_i u_i v_i^T \quad (5.49)$$

where σ_i are the singular values (strictly positive), $\{u_i\}$ and $\{v_i\}$ are the basis of $\mathfrak{R}(J)$ and the complementary space $\mathfrak{N}(J)$, see pages 167 et sqq.

The expression of the pseudo-inverse shows the strong influence of any small singular values, thus explaining the instability of the solution around the singularity, namely

$$J^\dagger = \sum_{i=1}^r \frac{1}{\sigma_i} v_i u_i^T \quad (5.50)$$

in which the minimum singular value approaches zero ($\sigma_i \rightarrow 0$) as a singular configuration is approached, i.e., at a singular configuration, becomes ill-conditioned.

In this context, the damping factor λ transforms the ill-behaved inverse term in (5.20) into a damped term converging smoothly to zero when the singular value becomes small, namely

$$J^{\dagger\lambda} = \sum_{i=1}^r \frac{\sigma_i}{\sigma_i^2 + \lambda^2} v_i u_i^T \quad (5.51)$$

This alternative pseudo-inverse provides continuous and feasible joint velocities even at the neighbourhood of singular points.

Choosing the value of λ is not straightforward. Several methods to determine λ have been proposed in the literature [31], most of them based on some Jacobian-dependent measure such as the Yoshikawa's manipulability value [26] or rate of change [32], as well as the smallest singular value of J [28] (although SVD has a high computational cost). All these methods act above a threshold value.

Finally, a discussion arises about the convenience of this *DLS-method* for postprocessing at milling tasks. It is easy to see that the constant λ introduces an algorithmic error, also away from a singular point. This error is introduced in terms of both direction and magnitude. DLS-defenders argue that, if the singular values are much larger than the damping factor (which is likely to be true far from singularities), then there is little difference between the two solutions, since in this case

$$\frac{\sigma_i}{\sigma_i^2 + \lambda^2} \approx \frac{1}{\sigma_i} \quad (5.52)$$

Nevertheless, for the scope of this thesis (devoted to milling tasks) and taking profit from the redundant additional joints, it has been taken as more efficient and precise the previously introduced method.

5.3.2. Solution of functionally-redundant manipulators (r_F)

As presented in the previous section, most researchers use the pseudo inverse J^{\dagger} and the projection onto the $\mathfrak{N}(J)$ of the manipulator to solve the inverse kinematic problem at redundant manipulators. Those RRS are of direct application on many cases.

However, in section 5.1.1. the concept of functional redundancy was introduced. It is more clearly highlighted in commonly used 6R manipulators in which J often is a full rank square matrix, i.e., its null space doesn't exist so the second term of (5.20), working on $\mathfrak{N}(J)$, can not be directly used. As a consequence, a new algorithms corresponding to the cases of full rank J have been recently developed in order to change (5.20) into an under-determined system. Two techniques are exposed: one by augmenting the dimension of joint-

rate (namely, *Virtual Joint Method*) and another one by reducing the dimension of the twist (namely, *Twist Decomposition Method*).

i) Virtual Joint Method (VJM)

Baron [21] proposed a joint limits avoidance strategy by adding a virtual joint around the symmetry axis of the tool (Figure 5.9), in order to obtain an under-determined linear algebraic system with at least one DOF of redundancy.

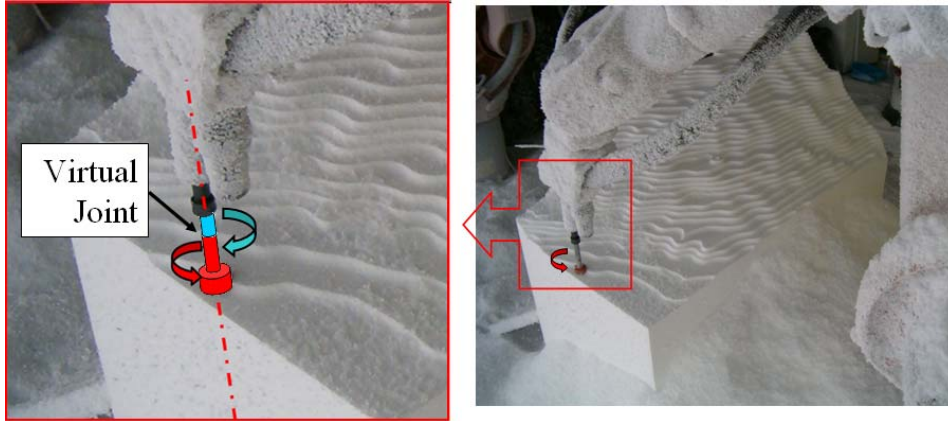


Figure 5.9. Additional virtual joint allowing a rotation around the symmetry axis of the tool.

Therefore, equation (5.20) can be rewritten as

$$\dot{q}_v = \tilde{J}_v^\dagger t + (I_{(n+1) \times (n+1)} - J_v^\dagger J_v) \cdot h \quad (5.53)$$

where J_v is an *augmented* Jacobian matrix by a virtual joint-rate \dot{q}_{n+1} , namely

$$\dot{q}_v = \begin{bmatrix} \dot{q}_i \\ \dot{q}_{n+1} \end{bmatrix} ; \quad i = 1, \dots, n \quad (5.54)$$

For example, in the explicit case of the IDF workcell, this additional virtual joint θ_7 implies having the joint-rate vector $\dot{q}_v = [\dot{\theta}_M, \dot{d}_L, \dot{\theta}_1, \dots, \dot{\theta}_6, \dot{\theta}_7]^T$.

ii) Twist Decomposition Method (TDM)

In general milling operations, the cutting tool has a symmetry axis. The tool holder can be rotated around this axis without affecting the task. This axis describes the geometry of the functional redundancy. In Figure 5.10, the unit vector e denotes the orientation of the symmetry axis along the milling tool.

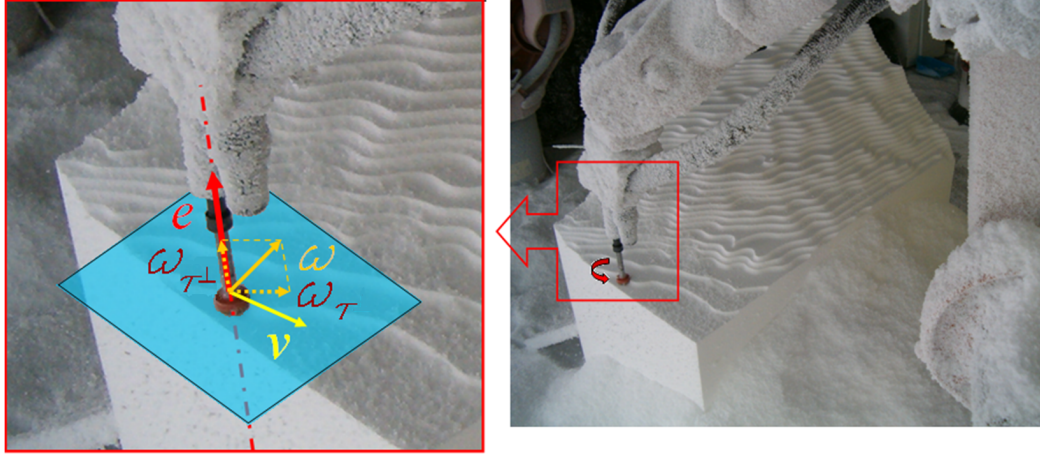


Figure 5.10. Decomposition of the angular velocity vector ω into two orthogonal parts: one lying on the task subspace (ω_τ) and another one lying on the orthogonal task subspace ($\omega_{\tau\perp}$).

Huo and Baron [22] decomposed the angular velocity in the twist vector ($t, 2 \times R^3$) of the minimum norm solution (first term of (5.20)) into two orthogonal subspaces, one in the task subspace and another lying into the orthogonal task subspace (Figure 5.10), namely

$$t = t_\tau + t_{\tau\perp} = T t + T^\perp t \quad (5.55)$$

where T is a twist projector matrix. The twist projectors for a general milling task can be defined as

$$T \equiv \begin{bmatrix} (I_{3 \times 3} - ee^T) & 0 \\ 0 & I_{3 \times 3} \end{bmatrix}; \quad T^\perp \equiv I_{6 \times 6} - T = \begin{bmatrix} ee^T & 0 \\ 0 & 0 \end{bmatrix} \quad (5.56)$$

Therefore, the first term of (5.20) can be rewritten as

$$\dot{q} = (J^\dagger T)t + J^\dagger (I_{6 \times 6} - T) \cdot t \quad (5.57)$$

The first part of (5.57) represents the relevant task displacement and the second part represents the redundant displacement, i.e. those components not important for the task. In fact, the author reconsidered these components as a way to deal with the redundancy by replacing t with an arbitrary vector h of joint spaces properly projected on the task space, allowing a secondary task to be satisfied. Thus, (5.57) is rewritten as

$$\dot{q} = (J^\dagger T)t + J^\dagger (I_{6 \times 6} - T) \cdot Jh \quad (5.58)$$

The TDM has great difference with the projection on $\aleph(J)$ (eq. (5.20)) on the theoretical base. Both of them consider a prior task and a secondary use of the redundancy, but the TDM projects the task from the robot base frame to the EE frame. Thus, the motion of the secondary task is always constant in the EE frame (the rotation around the symmetry axis of EE), while this secondary motion may or may not be constant in the base frame. Thus, TDM classifies the order of task priority in instantaneous EE frame instead of in robot base frame as by the previous null space approach, and the TDM was directly developed from the minimum-norm solution without considering the projection onto the null space of J .

5.3.3. Consideration for functionally-redundant (r_F) and intrinsically-redundant (r_I) manipulators

A final consideration must be done for the workcell studied in the present thesis, where both intrinsic and functional redundancies exist (Figure 5.4). Thus, it is very interesting to study the combination of the TDM and the projection on $\aleph(J)$ in order to take advantage from both types of redundancy, namely

$$\dot{q} = (J^\dagger T)t + J^\dagger (I - T) \cdot Jh_1 + (I - J^\dagger J) \cdot h_2 \quad (5.59)$$

where h_1 and h_2 are the two possible performance vectors for the redundant task. Clearly, h_1 is projected onto the functional redundancy, while h_2 is projected onto the intrinsic redundancy.

To the author's knowledge, a solution like (5.59) has not been studied. Clearly, (5.58) is a particular case of (5.59) with $h_2 = 0$.

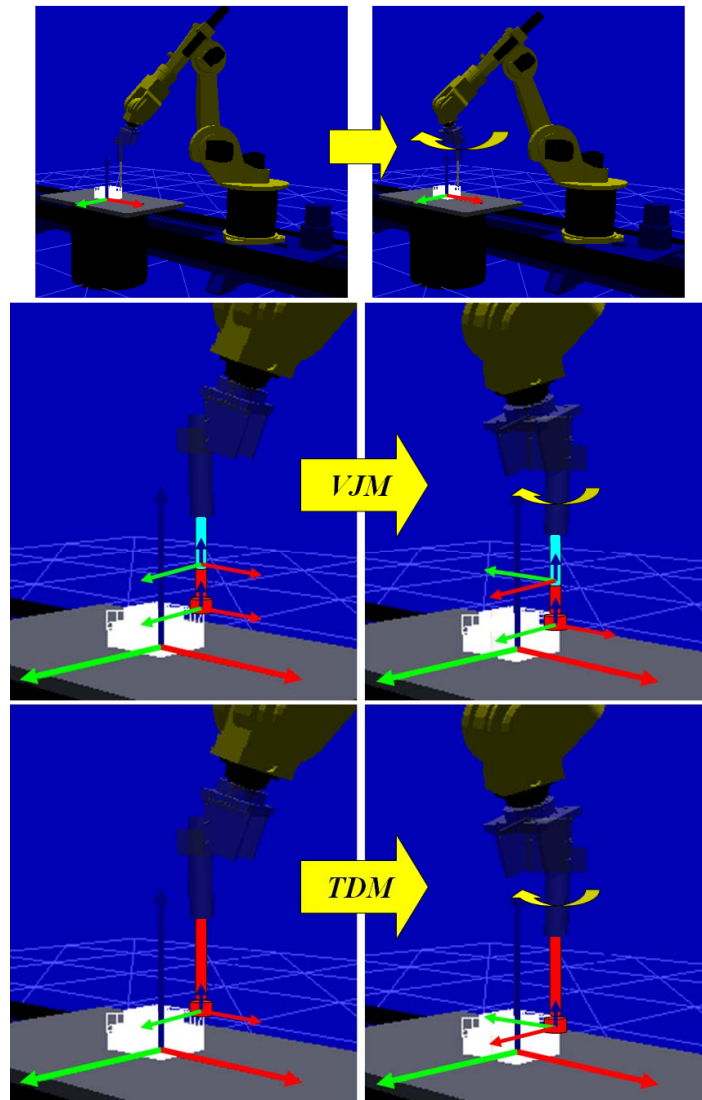


Figure 5.11. In the TDM, the motion of the secondary task is always constant in the EE frame (the rotation around the symmetry axis of EE).

Recalling Section 2.4.5., it is remarkable that the DH representation of the manipulators depends on the RRS selected, according to the significance explained in Section 5.3.2., by adding the additional joint or considering a fixed displacement up to the tool tip. For the VJM (Figure 5.12, left), and additional line is added in the DH-model (Table 5.1). The TDM uses the actual DH-model,

and therefore a final constant displacement matrix is required to know the position of the EE (Figure 5.12, right).

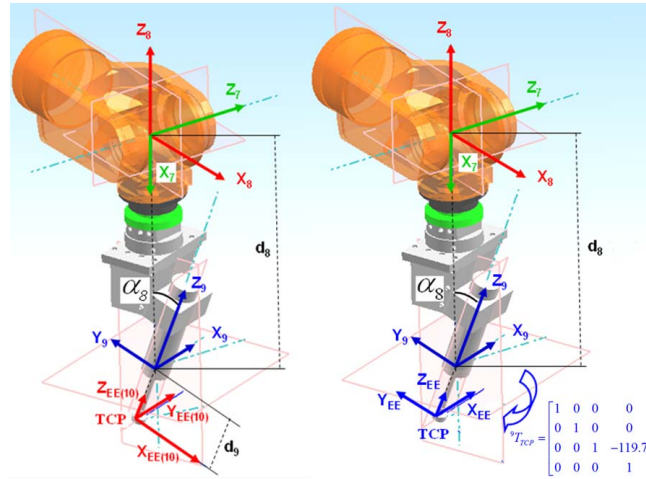


Figure 5.12. Comparison of the DH frame assignment for the VJM (left) and the TDM (right).

Link	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
1	$\pi/2$	803	θ_M	-305
2	$\pi/2$	0	0	d_L
3	$\pi/2$	300	θ_1	-675
4	0	650	θ_2	0
5	$\pi/2$	155	θ_3	0
6	$\pi/2$	0	θ_4	-600
7	$\pi/2$	0	θ_5	0
8	0.3564	0	θ_6	-443.4
TCP	0	0	$\theta_{7(VJM)}$	-119.7

Table 5.1. Table summarizing the parameters for both standard DH-models.

5.3.4. Redundant manipulator controlling process

The usual control flowchart to control an industrial redundant workcell is based on the experience of the workman in charge of the system. Such methods are directly deduced from the position IK problem that, in most industrial manipulators, can be solved rapidly (for example, by using geometric methods as those described in Chapter 2).

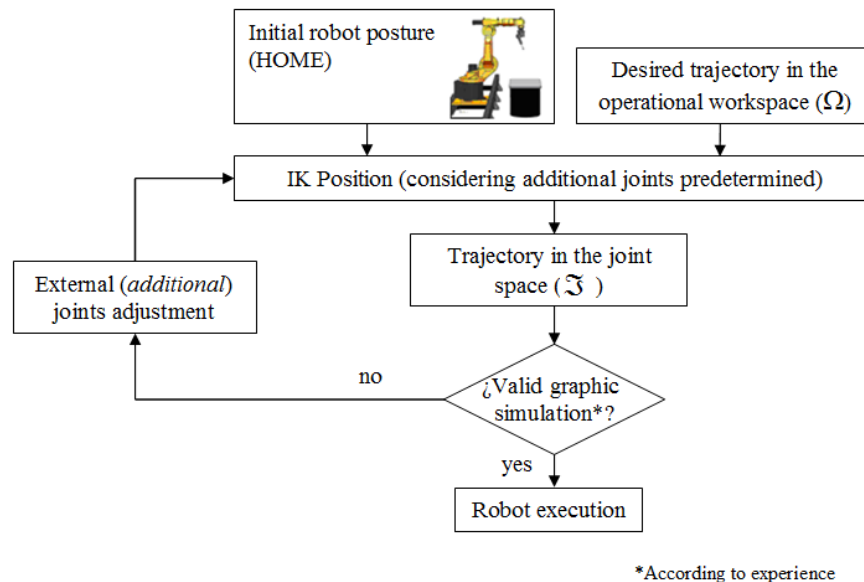


Figure 5.13. Usual flowchart for a redundant manipulator controlling process.

From the described RRS, another controlling process is deduced as shown in Figure 5.14, where the resolution is done at the joint rate level. In Chapter 6 several case studies will be developed following this reasoning. In the following, it can be interesting to highlight how the experience of the workman can be placed in such a flowchart. For that, a previous revision of the expert fuzzy systems is done in the next Section.

5.4. INTELLIGENT CONTROL IN REDUNDANCY RESOLUTION

Intelligent control is a new research direction making control systems to have higher degree of autonomy. The intelligent control methods applied on the Redundancy Resolution problems may include *fuzzy logics* (FL), *neural networks* (NN), and *genetic algorithms* (GAs).

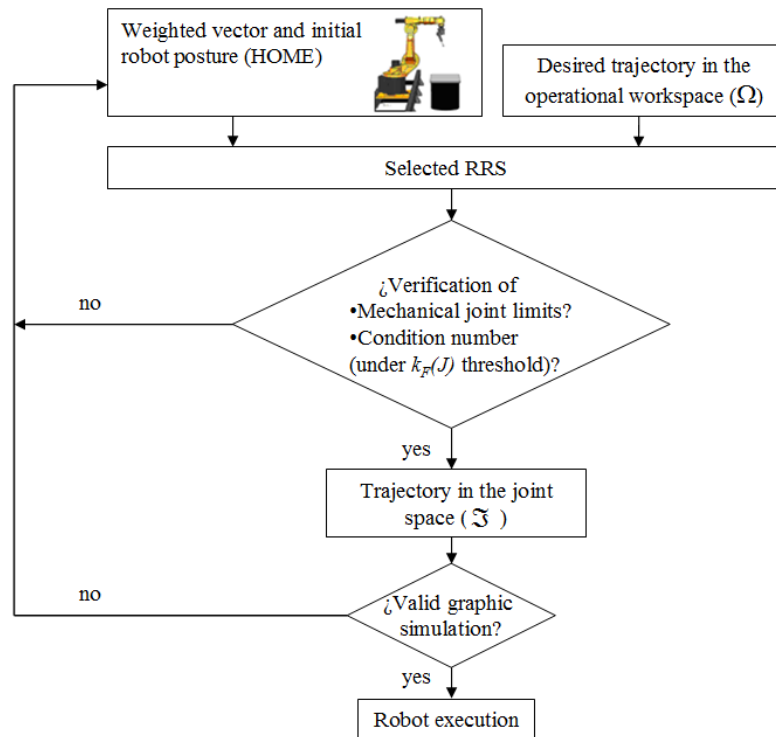


Figure 5.14. Proposed flowchart for a redundant manipulator controlling process.

There are many researches that want to replace some or all of the physical kinematic and dynamic modelling usually needed to implement conventional control techniques in robotics. With proper development, intelligent control approaches are supposed to have great potential for solving difficult control problems considering the complexity and computational cost of most of the mathematical models. Those goals are mainly pursued by *training and learning processes* of the NN or GAs, which then can be used to control the manipulator. This posture is reinforced with the argument not only of the *simplicity* but also of the *failure robustness* of these techniques just like in humans, i.e. not computing exact inverse kinematics but solving precise positioning from heuristics.

On the contrary, there are also many researchers who believe that this approach is not a good use of intelligent control algorithms. They argue that control techniques should keep as much physical modelling as possible, and let the intelligent control algorithm to handle the uncertainties or the unknown

physical phenomenon of the mechanical system at hand. This posture is reinforced with the support of the rapid development in computer technology.

Both perspectives find good reasons to keep going on, and in many cases they found each other in an intermediate point, taking mutual benefits. From this point of view, FL can find the reason for its use when tuning several aspects of the kinematic control which take into account an expert knowledge.

5.4.1. Fuzzy-Based Redundancy-Resolution Approaches

Considering the facts mentioned in the previous section, some efforts have been done to solve the inverse kinematics problem using FL methods [39][40][41][42][43].

Kim and Lee [39][40], and Xu and Nechyba [41], proposed two different approaches for fuzzifying the differential relationship between the differential twist and joint motions in the homogeneous solution of (5.20). In a similar way, Beheshti et al. [43] developed an optimized IK solving method through FL for real time applications, with a rule base indicating whenever one of the joint variables increases or decreases, which corresponding variables of the Cartesian space should increase or decrease. They applied this to a redundant 4R planar manipulator. Graca [44] proposed a FL algorithm for non-redundant robotic manipulators to track specified trajectories in Cartesian space. This algorithm consisted of treating the inverse of the Jacobian matrix as a matrix of fuzzy numbers, which was solved using fuzzy regression to obtain a fuzzy version of the Jacobian inverse matrix. In [45], he extended these inferencing techniques for optimizing the secondary task at redundant manipulators, i.e. for determining a fuzzy model for the performance index (h). For this, he constructed the rule base based on the desired subtask (singularity avoidance) by mean of observations on the determined symbolically Jacobian.

From a critical point of view, most of them do not propose a systematic method for generating and adjusting membership functions of fuzzy sets. The reason is that finding a fuzzy rule base for inverse kinematics of a redundant robot is a difficult task and the approaches exposed result as complex as the many corresponding analytical methods. Thus, creating and tuning these models are at the same level of complexity that other well-know mathematical models, even for the simplest cases such as planar manipulators that they deal with. From the author's point of view, it is also noteworthy the fact that many of these exclusive-fuzzy models lead efforts to develop the model but forgetting the real *raison d'être* of the fuzzy inference (i.e. the use of the expert knowledge of a skilled operator). In this sense, efforts can be senseless if they do not have a reasonably advantage in front of mathematical models or are founded in the observation of the same equations.

In the following Chapter, FL is included from two practical points of view: first, by implementing a control based on the IK positioning analysis of the workcell, and second by dynamically tuning several aspects of a RRS while taking into account an expert knowledge.

5.4.2. Fuzzy Logic Overview

A *Fuzzy Logic Controller* (FLC) is a controller that works internally with fuzzy variables. It comprises a *knowledge base* with definitions of *membership functions* and a *rule-base* (i.e. a set of *If-Then* statements), a *decision-making logic* (or *inference mechanism*), and *interfaces to and from* the physical world which allow the conversion from *crisp* values into *fuzzy* values and vice versa (i.e. *fuzzification* and *defuzzification* interfaces).

The *rule-base* contains a fuzzy logic quantification of the expert's linguistic description of how to achieve good control, while the *inference mechanism* (also known as *inference engine* or *inference module*) emulates the expert's decision making in interpreting and applying knowledge about how best to control a process. A block diagram of a fuzzy control system is shown in Figure 5.15.

A control cycle typically consists of taking process variables as input, converting them to fuzzy values (*fuzzification*), applying the input to the rule-base and deriving a fuzzy control action, converting this fuzzy control action to a crisp value (*defuzzification*), and giving this crisp value to the controlled process as control action.

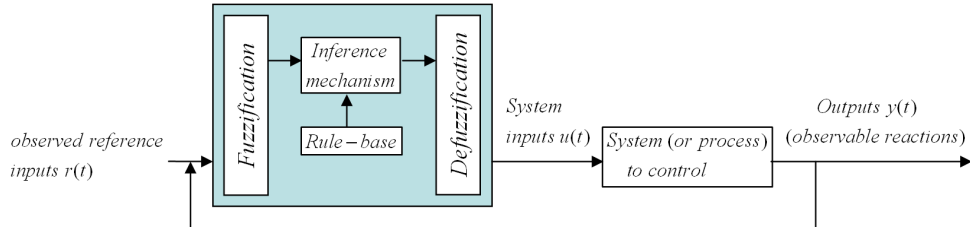


Figure 5.15. Block diagram of a fuzzy control. The inference mechanism interprets the values in the input vector and, based on some set of rules, assigns values to the system inputs.

The following subsections review briefly some of the concepts introduced, and the *inference process*.

i) Fuzzy sets and membership functions

Fuzzy logic starts with the concept of *fuzzy set*. A *fuzzy set* is a set without a clearly defined boundary. A fuzzy set admits the possibility of partial membership in it. In other words, this is the major advantage that fuzzy reasoning, namely the ability to reply to a yes-no question with a not-quite-yes-or-no answer. Humans do this kind of thing commonly, but it is a rather new trick for computers.

In fuzzy logics, the input space is sometimes referred to as the *universe of discourse*. Thus, if X is the universe of discourse and its elements are denoted by x , then a fuzzy set A in X is defined as a set of ordered pairs

$$A = \{x, \mu_A(x) \mid x \in X\} \quad (5.60)$$

where $\mu_A(x)$ is called the *membership function (MF)* of x in A . It is a curve that defines how each point in the *input space* is mapped to a membership value between 0 and 1. These membership functions are, in turn, built from several basic functions [54]: piecewise linear functions, Gaussian distribution functions, sigmoid curves and polynomial curves (see Figure 5.16). The simplest membership functions are formed using straight lines and, of these, the simplest is the *triangular* membership function.

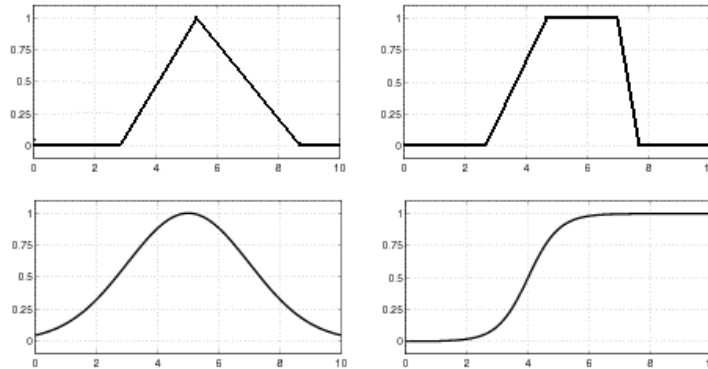


Figure 5.16. Types of Membership Functions: triangular MF, trapezoidal MF, Gaussian MF and Sigmoidal MF.

ii) Logical Operations

FL reasoning is a superset of standard *Boolean* logic. In other words, standard logical operations will hold by keeping the fuzzy values at their extremes of 1 (completely true), and 0 (completely false).

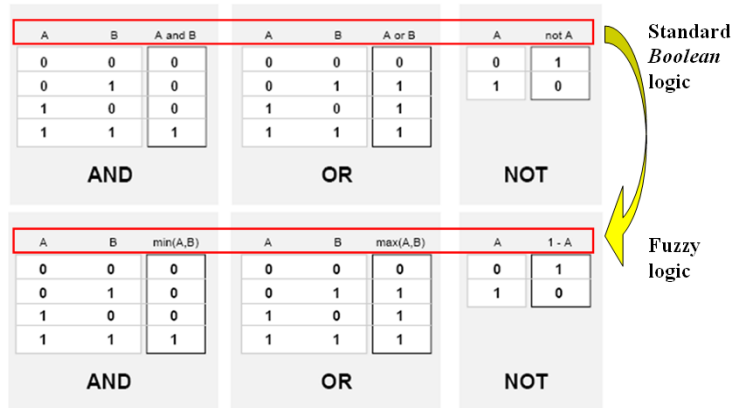


Figure 5.17. Standard truth tables adapted to FL reasoning: because there is a function behind the truth table rather than just the truth table itself, values between 1 and 0 can be considered now.

Due to the fact that in FL the *truth* of any statement is a matter of degree, the input values can be real numbers between 0 and 1 as previously stated. In this case, the $min(A,B)$ operation preserves the results of the A AND B truth table and also extend to all real numbers between 0 and 1. With the same reasoning, the OR operator can be replaced with the max function, so that A OR B becomes equivalent to $max(A,B)$. Finally, the operation NOT A becomes equivalent to the operation $1-A$ (Figure 5.17). In the Figure 5.18, the truth table is converted to a plot of two triangular fuzzy sets applied together to create one fuzzy set.

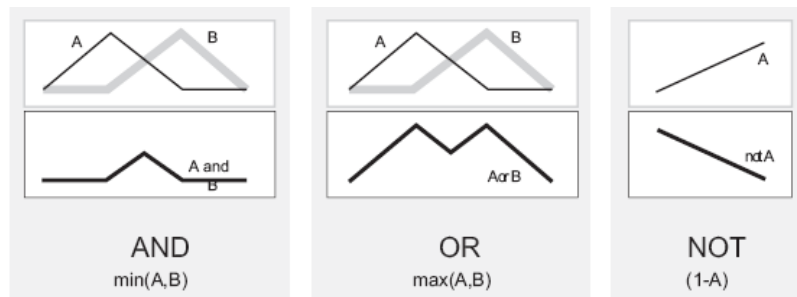


Figure 5.18. The upper fuzzy sets (A, B) are managed with the fuzzy operations defined, to get the result displayed below.

iii) If-Then Rules

The expert knowledge of the workman in charge of the system to manage (the robotic manipulator for the scope of this thesis) has to be considered in the design of the central core of the fuzzy controller.

To practical effects, this knowledge can be collected in *fuzzy association matrixes*, relating the control variables, the adjectives that describe those variables and the action associated to an expert managing. This information is necessary for the accomplishment of a set of rules, known as *rule-base*. In it, the stand alone controller will uphold the criterion for the decisions taken when managing the system.

These rules combine one or more fuzzy sets of entry (*antecedents*) and associate it with one or more output fuzzy sets (*consequents*). They are basically of the type of “*IF <antecedents> THEN <consequents>*”, being both fuzzy sets associated by fuzzy operators AND, OR, NOT.

iv) Fuzzy Inference Process

The process of *fuzzy inference* involves all of the pieces previously introduced. It is the process of formulating the mapping from a given input to an output using fuzzy logic.

Two types of fuzzy inference systems are commonly used, namely the *Mamdani-type* and *Sugeno-type* [50][51][54]. These two types of inference systems vary somewhat in the way outputs are determined. Ebrahim Mamdani’s fuzzy inference method is the most commonly seen fuzzy methodology [50], and it will be the inference method used in this thesis.

Fuzzy inference process comprises of five steps:

- *Fuzzification of the input variables*

The first step is to take the inputs and determine the degree to which they belong to each of the appropriate fuzzy sets via evaluation of the membership functions. The input is always a crisp numerical value limited to the universe of discourse of the input variable and the output is a fuzzy degree of membership in the qualifying linguistic set (always the interval between 0 and 1). In this manner, each input is fuzzified over all the qualifying membership functions required by the rules.

- *Application of the fuzzy operator (AND, OR, NOT) in the antecedent*

If there is only one part to the antecedent, then this is the degree of support for the rule. If there are multiple antecedent parts, after the inputs

are fuzzified, the fuzzy operator is applied to obtain one number that represents the result of the antecedents for that rule. This number is then applied to the output function.

- *Implication from the antecedent to the consequent*

The output fuzzy set is also represented by a membership function. The input for the implication process is the single number resulting from the previous step. If the antecedent is only partially true, (i.e., is assigned a value less than 1), then the output fuzzy set is truncated according to the implication method. Implication is implemented for each rule.

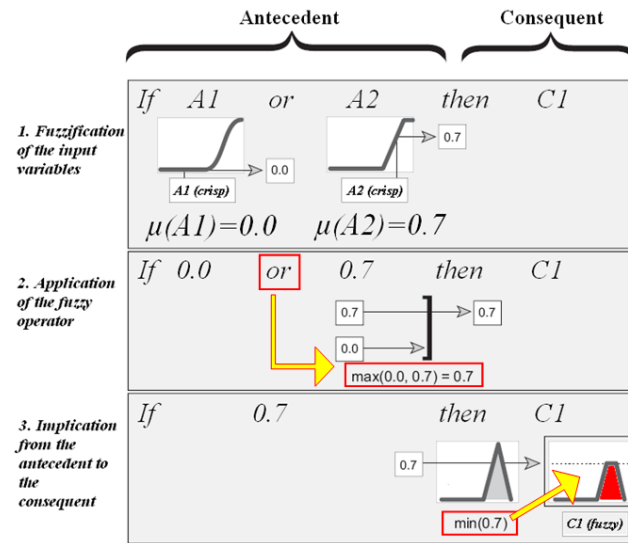


Figure 5.19. Fuzzification, application of the fuzzy operator (*OR*) and implication processes for a single *if-then* rule

- *Aggregation of the consequents across the rules*

In general, one rule alone is not effective. Two or more rules that can interact amongst them are needed. All rules are evaluated in parallel, the order of the rules is unimportant, and the output of each rule is a fuzzy set.

Because decisions are based on the evaluation of all of the rules, the rules must be combined in some manner in order to make a decision. Therefore, aggregation is the process by which the fuzzy sets that represent the outputs of each rule are combined into a single output fuzzy set for each output variable (Figure 5.20).

- *Defuzzification*

As much as fuzziness helps the rule evaluation during the intermediate steps, the final desired output for each variable is generally a single number. Therefore, from the previous aggregated output fuzzy set, the defuzzification process gives as a result a crisp value.

The most popular defuzzification method is the centroid calculation (Figure 5.20), which returns the center of area under the aggregated curve defining the single output fuzzy set.

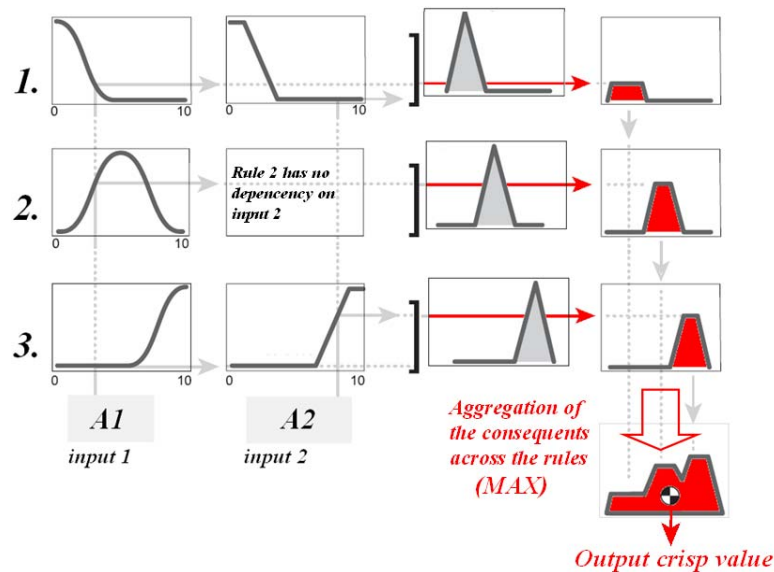


Figure 5.20. Aggregation of the consequents across the rules a single output fuzzy set, and final defuzzification by means of the centroid method. In summary, information flows through the fuzzy inference process as shown.

REFERENCES (Ch. 5)

- [1] Arenson, N., Angeles, J. and Slutski, L., Redundancy-resolution algorithms for isotropic robots, *Advances in Robot Kinematics: Analysis and Control*, pp. 425-434, 1998.
- [2] Siciliano, B., Solving manipulator redundancy with the augmented task space method using the constraint Jacobian transpose, *IEEE Intern. Conf. on Robotics and Automation, Tutorial M1*, pp. 5.1-5.8, 1992.
- [3] <http://mathworld.wolfram.com/NullSpace.html> (Accessed on 12-feb-2010).
- [4] Liégeois, A., Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms, *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 245-250, Mar. 1977.
- [5] Yoshikawa, T., Analysis and control of robot manipulators with redundancy, *Robotics Research: The First International Symposium*, pp. 735-747, 1984.
- [6] Nakamura, Y., *Advanced robotics: redundancy and optimization*, Addison-Wesley Pub. Co., Massachusetts, 337 pages, 1991.
- [7] Nakamura, Y. and Hanafusa, H., Task priority based redundancy control of robot manipulators, *MIT Press*, pp. 155-162, 1985.
- [8] H Hanafusa, T Yoshikawa, Y Nakamura; Analysis and control of articulated robot arms with redundancy, - 8th IFAC World Congress, 1981
- [9] Whitney, D.E., Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man-Machine Syst.*, vol. 10, no. 2, pp. 47-53. 1969.
- [10] Park, J., Chung, W. and Youm, Y., Weighted decomposition of kinematics and dynamics of kinematically redundant manipulators, *IEEE International Conference on Robotics and Automation, Vol. 1*, pp. 480-486, 1996.
- [11] Chang, T.-F. and Dubey, R.-V., A weighted least-norm solution based scheme for avoiding joints limits for redundant manipulators, *IEEE Trans. Robot. Automat.*, vol. 11, pp. 286-292, Apr. 1993.
- [12] Arenson, N., Angeles, J. and Slutski, L., Redundancy-resolution algorithms for isotropic robots, *Advances in Robot Kinematics: Analysis and Control*, pp. 425-434, 1998.
- [13] Yashi, O.S., and Ozgoren, K., Minimal joint motion optimization of manipulators with extra degrees of freedom, *Mechanism and Machine Theory*, Vol. 19, No. 3, pp. 325-330, 1984.
- [14] Angeles, J., Anderson, K. and Gosselin, C., An Orthogonal-Decomposition Algorithm for Constrained Least-Square Optimization, *ASME Robotics, Mechanisms and Machine Systems, Design Eng. Division, Vol. 2*, pp. 215-220, 1987
- [15] Baillieul, J., Avoiding obstacles and resolving kinematic redundancy, *IEEE International Conference on Robotics and Automation, Washington*, pp. 1698-1704, 1986.
- [16] Siciliano, B., Solving manipulator redundancy with the augmented task space method using the constraint Jacobian transpose, *IEEE Intern. Conf. on Robotics and Automation, Tutorial M1*, pp. 5.1-5.8, 1992.

- [17] Siciliano, B.; *Kinematic Control of Redundant Robot Manipulators: A Tutorial*; Journal of Intelligent and Robotic Systems 3: 201-212, Kluwer Academic Publishers 1990.
- [18] Arenson, N.; *REAL TIME REDUNDANCY-RESOLUTION SCHEMES FOR ROBOTIC MANIPULATORS*, Department of Mechanical Engineering-McGill University, Montréal, 1998
- [19] R.V. Patel and F. Shadpey; "Control of Redundant Robot Manipulators: Theory and Experiments"; Springer 2005, ISBN 10 3-540-25071-9
- [20] Baron L. and Huo L., "Inverse Kinematics of Functionally-Redundant Serial Manipulators: A Comparison Study"; 12th World Congress on the Theory of Machines and Mechanisms, Besancon, France, 18-21 juin 2007.
- [21] Baron L., "A joint-limits avoidance strategy for arc-welding robots", International Conference on Integrated Design and Manufacturing in Mechanical Engineering, Montreal, Canada, May 2000.
- [22] Huo L., and Baron, L., "Kinematic inversion of functionally-redundant serial manipulators: application to arc-welding", Transactions of the Canadian Society for Mechanical Engineering, 2005, Canada.
- [23] Huo L. and Baron L.; "The joint-limits and singularity avoidance in robotic welding"; *Industrial Robot: An International Journal* 35/5 pp 456-464 (2008)
- [24] Honegger, M. and Codourey, A., Redundancy resolution of a cartesian space operated heavy industrial manipulator, IEEE International Conference on Robotics and Automation, Vol. 3, pp. 2094-2098, 1998.
- [25] Baerlocher P., Boulic R.; "An inverse kinematics architecture enforcing an arbitrary number of strict priority levels", *The Visual Computer* 20:402-417, Springer (2004)
- [26] Nakamura, Y. and Hanafusa, H.: Inverse kinematic solutions with singularity robustness for robot manipulator control, *Trans. ASME, J. Dynamic Systems, Measurement and Control* 108, 163-171 (1986).
- [27] Baerlocher P, Boulic R; Task-priority formulations for the kinematic control of highly redundant articulated structures. Proc of IEEE IROS 98, Victoria, BC, pp 323-329, (1998).
- [28] Maciejewski A. A., Klein C. A., "Numerical filtering for the operation of robotic manipulators through kinematically singular configurations," *Journal of Robotic Systems*, vol. 5, no. 6, pp. 527-552, (1988).
- [29] Buss S.R.; "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods", Department of Mathematics, Department of Mathematics (October 2009). Unpublished.
- [30] C. W. Wampler, Manipulator inverse kinematic solutions based on vector formulations and damped least squares methods, *IEEE Transactions on Systems, Man, and Cybernetics*, 16, pp. 93-101, (1986).
- [31] DEO A. S., WALKER I. D.; "Overview of Damped Least-Squares Methods for Inverse Kinematics of Robot Manipulators", *Journal of Intelligent and Robotic Systems* 14: 43-68, (1995).
- [32] Kelmar, L. and Khosla, E K.: Automatic generation of kinematics for a reconfigurable modular manipulator system, in Proc. 1988 IEEE Conf. Robotics and Automation, Philadelphia, PA, 1988, pp. 663-668.
- [33] Chiaverini S., Oriolo G., Walker I. D.; "Kinematically Redundant Manipulators". *Springer Handbook of Robotics*, pp. 245-268, (2008).

- [34] TISIUS M., PRYOR M., KAPOOR Ch., TESAR D.; "An Empirical Approach to Performance Criteria for Manipulation", *Journal of mechanisms and robotics* vol. 1, no3 (2009).
- [35] McGhee, S., Chan, T., and Dubey, R., "Probability-Based Weighting of Performance Criteria for Redundant Manipulators," *Proceedings of IEEE International Conference on Robotics and Automation*, San Diego, CA, pp. 1987–1984, (1994)
- [36] Marani G., Kim J., Yuhl J., Chung W. K.; "A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators"; *IEEE International Conference on Robotics and Automation*, pp. 1973-8, (2002).
- [37] KHOUKHI A., BARON L., BALAZINSKI M., "PLANIFICATION MULTI-OBJECTIFS DE TRAJECTOIRE DES ROBOTS REDONDANTS PAR LAGRANGIEN AUGMENTÉ ET GRADIENT PROJETÉ", *TRANSACTIONS-CANADIAN SOCIETY FOR MECHANICAL ENGINEERING*, VOL 31; NUMB 4, pages 391-406 (2007)
- [38] KEMÉNY Z.; "MAPPING, DETECTION AND HANDLING OF SINGULARITIES FOR KINEMATICALLY REDUNDANT SERIAL MANIPULATORS"; *PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 46, NO. 1, PP. 29–45* (2002)
- [39] Kim, Sung-Woo and Lee, Ju-Jang, Resolved motion rate control of redundant robots using fuzzy logic, *IEEE International Conference on Fuzzy Systems*, pp. 333-338, 1993.
- [40] Kim, Sung-Woo, Lee, Ju-Jang, and Sugisaka, Masanori, Inverse kinematics solution based on fuzzy logic for redundant manipulators, *Int. Conf. Intell. Rob. Syst.*, pp. 904-910, 1993.
- [41] Xu, Y. and Nechyba, M. C., Fuzzy inverse kinematic mapping: rule generation, efficiency, and implementation, *Int. Conf. Intell. Rob. Syst.*, pp. 911-918, 1993.
- [42] Ramos, M.C. and Koivo, A.J., Fuzzy logic-based optimization for redundant manipulators, *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 4, pp. 498-509, 2002.
- [43] Beheshti M., Tehrani A., Ghanbari B.; "An Optimized Adaptive Fuzzy Inverse kinematics Solution for Redundant Manipulators"; *Proceedings of the 2003 IEEE Int. Symposium on Intelligent Control*, Houston, Texas; pp. 924-929, (2003)
- [44] Graca, Randy A. and Gu, You liang, A fuzzy learning algorithm for kinematic control of a robotic system, *Proceedings of the IEEE Conference on Decision and Control*, Vol. 2, pp. 1274-1279, 1993.
- [45] Graca, Randy A. and Gu, You-Liang, Application of the fuzzy learning algorithm to kinematic control of a redundant manipulator with subtask optimization, *IEEE International Conference on Fuzzy Systems*, Vol. 2, pp. 843-848, (1994).
- [46] Pieper, D.L., *The Kinematics of Manipulators under Computer Control*, Ph.D. thesis, Stanford University, 1968.
- [47] Dahlquist, G. and Björck, A., *Numerical Methods*, Prentics-Hall, Englewood Cliffs, New Jersey, 573 pages, 1974.
- [48] Angeles, J., *Fundamentals of robotic mechanical systems: theory, methods and algorithms*, Springer, New York, 521 pages, 2003.
- [49] Whitney, D.E., "The mathematics of coordinated control of prosthetic arms and manipulator", *ASME J. Dynamics Systems, Measurement and Control*, Vol. 94, No. 4, pp. 303-309, 1972.

-
- [50] Mamdani, E.H. and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, Vol. 7, No. 1, pp. 1-13, 1975.
- [51] Sugeno, M., *Industrial applications of fuzzy control*, Elsevier Science Pub. Co., 1985.
- [52] Passino K. M., Yurkovich S.; "Fuzzy Control", Addison Wesley Longman, Inc., ISBN 0-201-18074-X (1998).
- [53] Jantzen J., Tutorial on Fuzzy Logic; Technical University of Denmark, Department of Automation, Lyngby, DENMARK., Tech. report no 98-E 868, 19 Aug 1998.
- [54] Reyer, R., Nicolas, C. F.; *Sistemas de control basados en lógica borrosa : fuzzy control*, Omron Electronics, 1995. ISBN 8492032626
- [55] Roger Jang J.S., Gulley, N.; *Fuzzy Logic Toolbox: User's Guide; Revised for Version 2.2.7 (Release 2008)*, The MathWorks, Inc. 2008
- [56] Park, J., Choi Y., Chung W.K., Youm Y.; "Multiple Tasks Kinematics Using Weighted Pseudo-Inverse for Kinematically Redundant Manipulators"; *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, pp. 4041-4048, Seoul, Korea; May 21-26, 2001

CHAPTER 6

ANALYSIS AND RESULTS

*“I'm so happy 'cause today / I
found my friends / They're in my head /
I'm so ugly, that's okay / 'Cause so are
you / Broke our mirrors / Sunday morning
is everyday / For all I care / And I'm not
scared / Light my candles, in a daze /
'Cause I've found God”*

Kurt Cobain

CHAPTER 6. ANALYSIS AND RESULTS

6.1. INTRODUCTION

The main difficulty of postprocessing a toolpath generated by a CAM platform for a complex robotic cell focuses on the treatment to give to the redundant joints in order to avoid singularities and limits of range (Figure 6.1). With the inherent redundancy stated previously, the aim is to reach the successive positions of the toolpath in the Cartesian Operational space Ω following a criterion of precision and economy in the whole motion of the manipulator. This raises two differentiated tasks referring to both *tool pose* and *manipulator posture*:

- Translation of the *tool pose* information generated by the CAM platform in agreement with the requirements of the robot language.
- Kinematics analysis of the robotic cell for the required the cutting *tool pose* at Ω , in order to include the treatment of the *manipulator posture* at \mathfrak{S} with the additional joints.

Chapter 6 is focused on the implementation of a control system for the redundant workcell previously described in Chapter 2, by following the algorithms and methods described in Chapter 5.

Traditionally, several robot manufacturers solve the problem by means of graphic simulator interfaces as an intermediate step between the CAM platform and the robot execution. An *expert* operator fixes the additional joints and checks the motions of the robot during the planned tracking, in order to know if a limit of range or a singular configuration is reached at any point. Figure 6.1 shows two singular configurations concerning the milling processes on the rotary table [1], and also different practical expert solutions to avoid both the widespread singularity (by means of a linear axis displacement) and the wrist singularity (by means of a table rotation).

As first attempt, it may be desirable the employment of such a fast and robust methodology that emulates the expert reasoning like the *fuzzy control* [2][3]. It is exposed in the following Section. Nevertheless, due to the fact that the positioning problem is highly non-linear, it results cumbersome to deal with. Thus, a second attempt is described in Section 6.3. with the RRS described in Chapter 5, namely the VJM and the TDM, which are compared. Additionally, some improvements on these methods are done, also taking benefit of the *fuzzy*

logic (see Section 6.3.2.). It can be applied on some tips that normally are subjected to author's estimation.

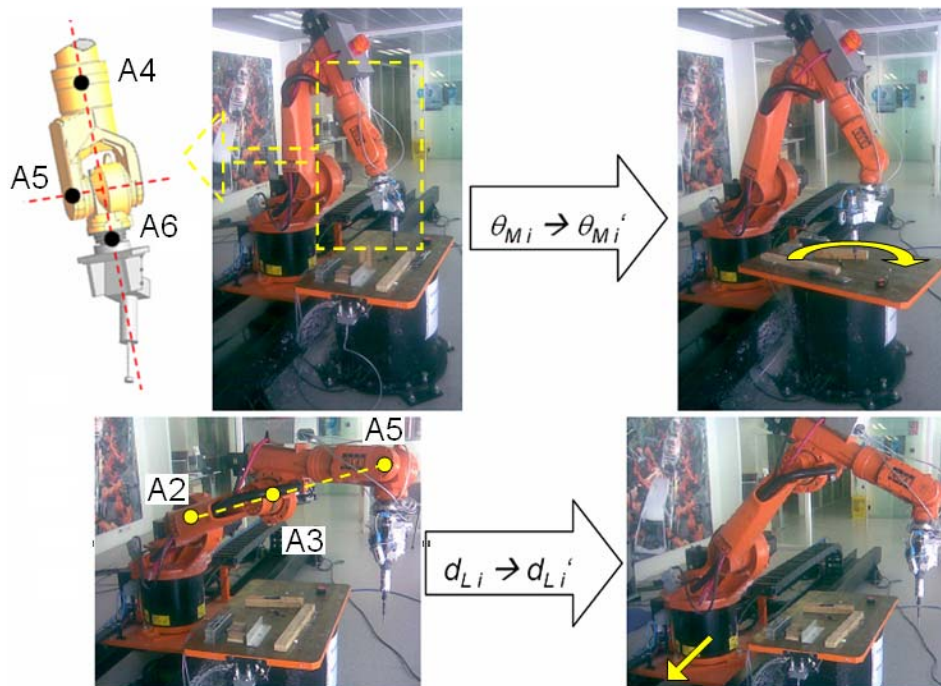


Figure 6.1. Wrist singularity (top) and widespread position singularity (bottom) concerning the milling processes on the rotary table.

6.2. FUZZY LOGIC FOR IK POSITIONING (IKP) PROBLEM

To practical effects, when the control the KUKA™ *industrial* workcell is carried out by an expert operator, both additional joints (external linear track E1 and rotary table E2) are requested only in case of avoiding singularities or limits of range in the chain A1-A6 (Figure 6.1). This *main* chain must be understand as the part of the robotic system devoted to locate the tool in Ω . Both additional joints should be placed taking profit from the fact that, after fixing whatever optimal valid values of E1 (d_L) and E2 (θ_M) to reach the proper *tool pose*, the path tracking is not affected since the controller adapts the A1-A6 values (namely, $\{\theta_1, \dots, \theta_6\}$).

Nevertheless, due to the KRC2 characteristics and the entry data structure required for CP tracking (Section 3.1.2.), the IKP analysis of the manipulator is necessary for the *expert fuzzy* evaluation of the optimal location of

E1 and E2. To make feasible the IKP resolution of this redundant system, the previous E1 and E2 joint values were considered to be known, being d_L and θ_M , respectively (see Chapter 2)¹. It is logical as it allows rapidly taking the successive toolpath coordinates at Ω to \mathfrak{S} , where the *convenience* of a new *robot posture* may be assessed. In other words, a fuzzy engine can just *decide* if the current *posture* is convenient for the milling operation or, on the contrary, if it is better to relocate the robot (by means of the linear track) or the workpiece (with a table rotation).

In the following paragraphs, the development of the structure for the controller will be explained, whereas its implementation inside NX's postprocessor will be developed in the following section. This controller will be implemented with Matlab (The MathWorks, Inc.) by means of its *Fuzzy Logic Toolbox* [6].

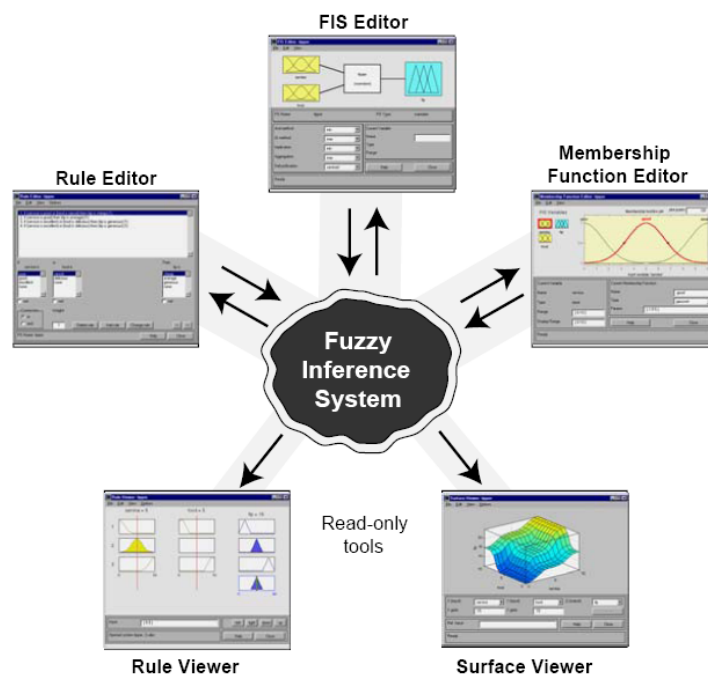


Figure 6.2. Overview of the Matlab's *Fuzzy Logic Toolbox*, which allows the design and testing of a fuzzy controller.

¹ To face the problem of the existence of infinite possible solutions and with the aim of automate the problem, it seems to be logical and profitable to take a solution near to the previous one [4]. Therefore, it is natural to store the previous position to minimize displacements of the additional joints E1 and E2.

This toolbox generates a *.fis* file (namely, *fuzzy inference system* file) in which the characteristics of the fuzzy model are saved. In addition, two C++ files (*fismain.c* and *fis.c*) are provided as the source codes to implement a stand-alone fuzzy inference engine.

6.2.1. Development of the fuzzy controller

This section approaches the design of the brain that controls the automated cell, that is, the *fuzzy controller*. It is expected to have the capacity to take decisions and therefore to govern the robotic mechanism. As the fuzzy logic admits different degrees of membership of any information inside the diffuse sets (see Chapter 5), a fuzzy controller can analogously determine different degrees of actuation of the robotic system.

The flow of the information of the joints in the *expert system* proposed is shown in Figure 6.3. The point of departure for all milling processes is a known posture of the workcell (HOME), which is prior and common to the execution of any subsequent program. From this posture, and depending on the point in the Cartesian space to which the TCP of the tool must come, the positioning of the external axes is reconsidered on the basis of the programmed *fuzzy controller*.

i) Variable definition

The output variables of the fuzzy system are clearly identified by the data structure required by the KRC2 controller: $\Delta\theta_M$ and Δd_L , that is, the incremental values to adequate the position of E1 and E2 (in addition to the tool pose that is a unavoidable data given by the CAM)².

On the basis of those data and after studying which robot joints are more affected applying the previous output variables, the input variables are structured. These variables are strongly dependent on the architecture of the workcell and the shape of the tool holder (Chapter 2). Two input variables are defined: θ_3 and θ_5 , both directly concerning the singular configurations that affect the operability of the arm on the table. They can be obtained with the previous IKP geometric computation described in Chapter 2.

² For CP commands, with the six coordinates of the tool pose and the values of the external joints (E1 and E2) the KRC2 internally solves the posture of the manipulator (Chapter 2).

ii) Clusterization of input and output spaces

As much the input space as the output spaces can be divided in five triangular clusters. This type of clusterization presents a major simplicity in its representation, managing and evaluation (Figure 6.4). The number of clusters is related to the linguistic etiquettes assigned, according to the experience.

It can be noticed that the functions neither are equidistant nor have identical form. It depends on the expected reactions, such as a steady state in intermediate values in the case of θ_3 but a greater displacement if θ_5 is almost aligned (0°). In case of θ_3 , it may be convenient the existence of a few dead zone without overlapping in which only a set for universe, and in consequence an alone rule, would be activated.

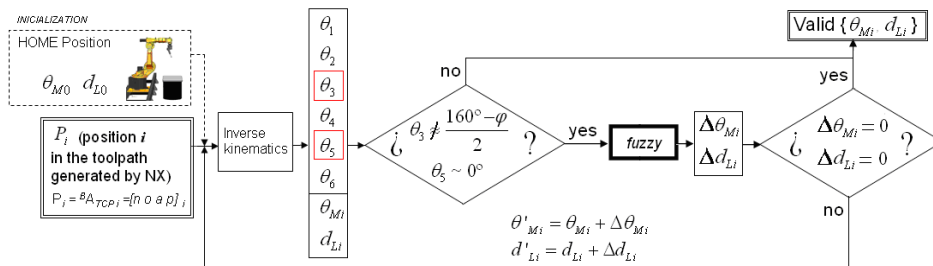


Figure 6.3. Flow of the heuristic reasoning in the control of the automated cell and its interaction with the expert system implemented in NX™.

iii) Fuzzification of the input variables

Figure 6.4 represents graphically the fuzzification process. The inputs to the fuzzy controller are discrete values in the range of the mechanical joint limits. For each of the input variables θ_3 and θ_5 , the value is compared with its respective space and associated with a cluster. Subsequently, the controller calculates the membership value μ of every input in each of the clusters being affected. Due to the particular partition of the spaces shown, the variable only could belong to one or two clusters, resulting in only one or two membership functions for space.

iv) Knowledge base

As justified in the previous Chapter 5, the *expert knowledge* of the operator in charge of the robots can be collected in two *fuzzy association*

matrixes (Table 6.1 and Table 6.2), which are necessary for the accomplishment of the *rule-base*. With two outputs it has been necessary to establish control for each one (i.e., two *association matrixes*), even when they have relative dependence on the same inputs.

It is easy to detect the vicinity of a wrist singularity configuration by means of the value of θ_5 . The expert system is supposed to reconsider this value when it is near to zero.

To practical effects, θ_3 should be considered when its value is near to provoke an extended position singularity, i.e. if $\gamma = \pi$ (Figure 2.24), so that $\theta_3 = \varphi$. In this case, the optimal position is considered to be in the intermediate position between the extended position and the closer limit to the robot base (namely, when $\theta_3 = 160^\circ$).

$\Delta\theta_M$		θ_3				
		VC	C	R	O	VO
θ_5	VN	Q	Q	Q	Q	Q
	N	Q	Q	Q	Q	Q
	ALI	CCW	CCW	CW	CW	VCW
	P	Q	Q	Q	Q	Q
	VP	Q	Q	Q	Q	Q

Table 6.1. Knowledge base for $\Delta\theta_M$

Δd_L		θ_3				
		VC	C	R	O	VO
θ_5	VN	MFA	FA	Q	AP	MAP
	N	MFA	FA	Q	AP	MAP
	ALI	MFA	FA	Q	AP	MAP
	P	MFA	FA	Q	AP	MAP
	VP	MFA	FA	Q	AP	MAP

Table 6.2. Knowledge base for Δd_L

a. Abbreviations. ((V)C=(Very) Closed, R=Relaxed, (V)O=(Very) Opened, (V)N=(Very) Negative, ALI=Aligned, (V)P=(Very) Positive, (M)AP=(Much) Approach, Q=Quiet, (M)FA=(Much) Far Away, (V)CCW=(Very) Counter-Clockwise, (V)CW=(Very) Clockwise.

v) Inference engine

The process of inference used is that of Minimum-Maximum (*Mandami*), as described in Chapter 5. The result of the *fuzzification* gives certain membership values in different clusters at every space of entry (θ_3 and θ_5). These values (considered as the *antecedents*) are leaked on the

base of rules to know in which clusters of the output spaces take place the *consequents*. The membership value inherited to the output clusters of every fulfilled rule is the *minimal* membership value of the clusters of the input spaces (*antecedents*) involved in that rule (Figure 6.4, in red).

Finally, it is necessary to *compose* the output polygon (Figure 6.4, in blue), which reflects the membership values in the clusters of the output spaces ($\Delta\theta_M$ and Δd_L) along the set of fulfilled rules. For this, it is necessary to review every cluster of the output spaces at all applied rules where the *consequents* coincide, taking the *maximum* membership value which it presents at any rule.

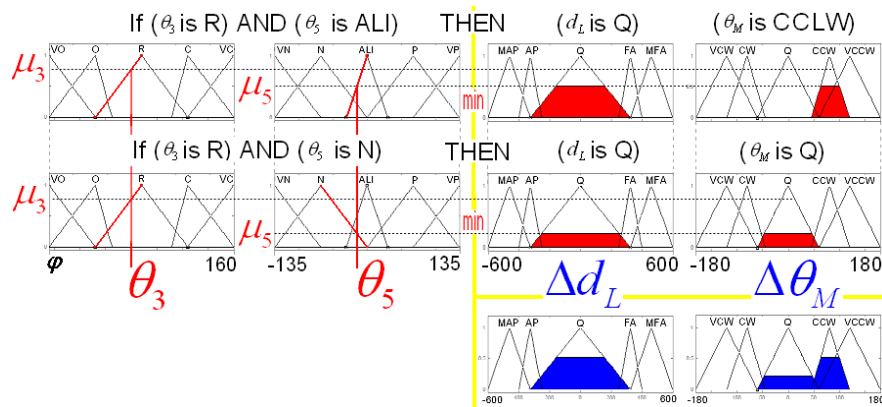


Figure 6.4. Min-Max inference process for two rules: two discrete input values (θ_3, θ_5) are fuzzified (μ_3, μ_5) by the corresponding clusters involved in both rule. The minimum degree of membership in each case is taken as output membership value in the implied output clusters (red), and then aggregated (blue) into a single fuzzy set for the overall output.

vi) Defuzzification

As final part of the fuzzy process, the *defuzzification* is carried out by means of the *centroid* method applied to the figure that results from the prior *composition*, as described in Chapter 5 (Figure 6.5). The resultant position on the horizontal axis is the defuzzified value of the output variable (*crisp output*).

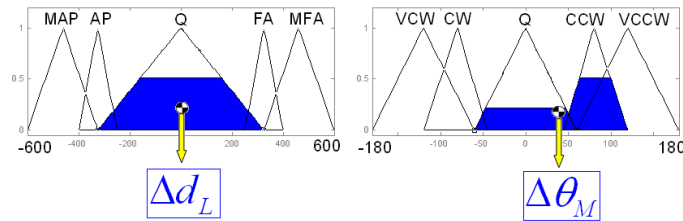


Figure 6.5. The *centroid* calculation returns the center of area under the aggregated curve as crisp output value.

vii) Interactivity with fuzzy controller module.

As shown in Figure 6.6 (whose origins are in Figure 4.20.), the *Event Handler* has been programmed to pass the next desired *tool pose* coordinates at {B} and the current external joint values (i.e., from the current position prior to any relocation of the tool possibly requiring a movement of the external joints) to a C++ programmed module. It makes two tasks:

- The *inverse kinematics calculation*, to obtain the $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$ joint values, corresponding to the next desired position with the current d_L and θ_M values.
- The *fuzzy control*: with the calculated joint values, the embedded fuzzy inference engine reads the *.fis* model and returns the estimated $\Delta\theta_M$ and Δd_L to reach the desired position of the TCP with an optimal configuration.

These values are checked as valid by means of an iterative cycle (Figure 6.3), and then returned to the *Event Handler*, which passes the final values to the *Definition File* for its publication.

It is important to note that, to practical effects and due to the architecture of the KRC2 controller, the values passed are the same *tool pose* coordinates at {B} but with the recalculated values of the additional external joints E1 and E2, as justified in the previous section i.1) and [5].

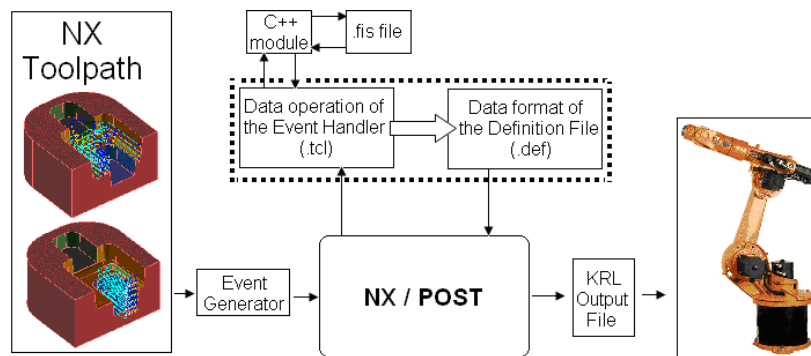


Figure 6.6. Integrated postprocessing in NX. The *Definition File* and the *Event Handler* are programmed in TCL to adapt NX's CAM to the KUKA KRC2 controller. The *Event Handler* is able to interact with executable modules programmed in C++.

6.2.2. Analysis and results

After the implementation of the postprocessor, further simulation was carried out in order to verify if it realizes satisfactory control actions. This simulation was first run with Matlab's toolbox *Hemero* [10], and then the result was compared with the real robot movement.

Figure 6.7 and Table 6.3 illustrate the behaviour of the previously described controller when an adjustment of θ_3 is obviously required (case A), or both θ_3 and θ_5 (case B).

	CASE A		CASE B	
P_i	[0, 0, 10]		[-75, 10, 0]	
	Before	After	Before	After
d_L	-2970	-2603	-2800	-2346
θ_M	0	0	-75	-51
θ_1	13.65	33.21	25.93	50.24
θ_2	-65.94	-55.72	-84.66	-59.46
θ_3	129.02	111.44	157.66	119.32
θ_4	-19.86	-7.79	-67.72	-26.96
θ_5	53.85	63.31	31.99	54.07
θ_6	33.18	42.42	18.09	23.87

Table 6.3. Case studied

As it can be appreciated, in both cases the result was the desirable with the first iteration, obtaining the same values for θ_M and d_L in a second calculation with the values previously obtained. In this sense, the implemented fuzzy control works as desired in terms of rapidity in the response.

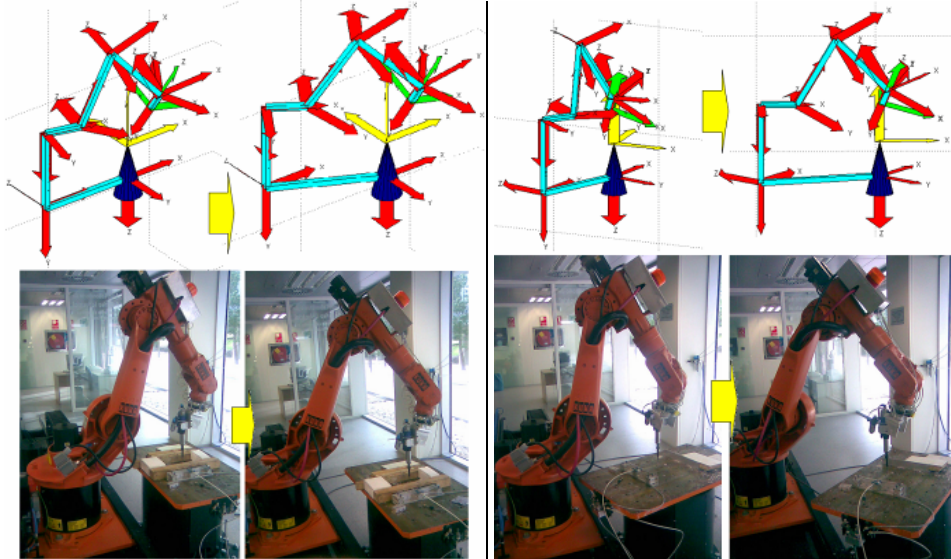


Figure 6.7. Matlab simulation of the readjustment of the workcell after the actuation of the implemented fuzzy controller for Case A (left) and B (right).

However, it is easy to highlight some *limitations* for this implementation. First, the rule base should be wider enough to consider all the possible poses of the tool when milling on the table. This is the reason why it has been only performed successfully for a 3-axis milling (i.e. with a constant tool orientation like shown in Figure 6.7). Even in this case, the configuration of a rule base is cumbersome due to the high non-linearity of this system which makes it unpredictable in a certain way and in some situations, even for an experienced workman. For that reason, the previous fuzzy implementation may be re-considered for its use at the rate level, where the problem becomes linear.

6.3. IK PROBLEM IMPLEMENTATION AT RATE LEVEL

6.3.1. Discussion

In the previous Section, the IK problem was managed at the position level. Clearly, despite the obtained results are discussed in a later section, that

problem was highly non-linear. Thus, it can be deduced that despite the application of *fuzzy logic* to deal with these non-linearities, some difficulties will appear when configuring such a wide rule-base foreseeing all possible situations in a complex milling task. Moreover, it can be criticized the employment of the fuzzy logic itself when some other control methods for redundant manipulators have been developed, as shown in Chapter 5.

In this sense, the IK problem at joint-rate level was introduced at Section 2.2.2-ii). In addition, this problem was highlighted for the case of redundant manipulators, by means of eq. (2.13). Nevertheless, it was in Section 5.3.1 when several methods derived from this equation were introduced.

It is important to note that the great acceptance of these RRS founded on the joint-rate level is justified by the fact that the non-linear position problem is converted to a linear problem at the velocity level. In fact, in Section 2.2.2-iii), Whitney's geometric Jacobian matrix was described as the mapping between the joint rates and the twist of the tool tip. From a practical point of view, this Jacobian can be easily evaluated numerically for each given posture of the robot.

This section deals with the control of the workcell with some of these methods by comparing two of them, namely the TDM and the VJM described in Chapter 5. The implementation by using the Householder-Reflections will be taken into account in both cases as described in Section 5.3.1-iii). As justified there, both methods are more suitable for milling applications than any other based on the DLS-inverse, since the position and orientation of the TCP are highly compulsory³.

Finally, several considerations on the utility of fuzzy logics for the assignment of the *performance vector* h are done. Thus, some valid improvements for both methods are implemented and tested.

6.3.2. TDM and VJM test implementation

i) Method

A challenging 5-axis milling was done as test to compare the performance of the TDM and the VJM. It consists of a spherical shape to be milled through a continuous spiral path. As shown in Figure 6.8, the symmetry axis of the tool (in red) is required to point constantly the center of the sphere, while the successive TCP positions are depicted in blue. This shape is supposed to be located into the manipulator's workspace on the

³ As exposed in Chapter 5, the damping factor (λ) would introduce an algorithmic error, also away from a singular point, in terms of both direction and magnitude.

rotary table, namely the base $\{B\}$. For the test, the center has been somehow located at an arbitrary point in the workspace, with the coordinates

$$C = \{100, 200, 250\} \quad (mm) \quad (6.1)$$

and the radius of the sphere has been set in $R=150$ mm. For the scope of this thesis, the tool holder designed at the IDF for milling purposes (described at Figure 2.28 and Table 2.5) is considered in this comparison, as depicted in Figure 6.8.

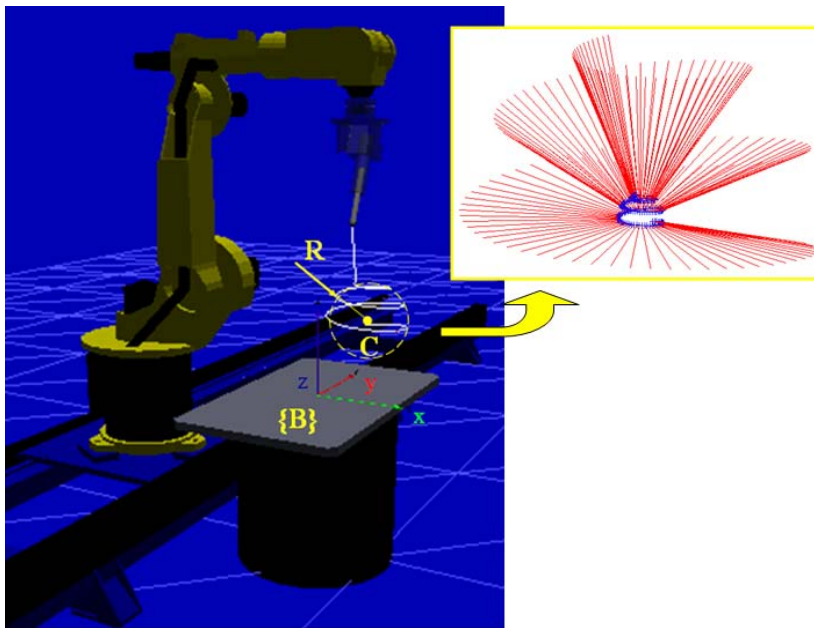


Figure 6.8. Workcell at HOME posture and main parameters of the experimental toolpath.

It is important to remark that this test is highly demanding. In fact, common milling paths are composed of relatively short trajectories which are concatenated by means of other *void motions* (i.e. motions with the tool not touching the workpiece) in which the robot or machine tool is relocated. Instead, we are aiming for this relocation of the additional external joints meanwhile a long and challenging path-tracking is being followed.

This test has been programmed in Matlab for both TDM and VJM methods, within the same suppositions. The trajectory data, generated with NX^{TM} , is kept as T_{CAM} . Starting from the HOME posture shown in Figure 6.8 ($q_0 = [+π, 0, +π, -π/2, 0, 0, +π/2, 0]^T$ rad) and with the DH models of both the KR15/2 manipulator (DH-KR15/2, Table 2.2) and the complete workcell (DH-Workcell, Table 2.3) at hand, the programmed algorithm is summarized as follows on the basis of (5.16):

$$\begin{aligned}
& 1) \quad q \leftarrow q_0 \\
& \text{for } (\sim \text{each } i\text{-point of the trajectory, } T_{CAM}(i)) \\
& \quad 2) \quad \{p_d, Q_d\} \leftarrow T_{CAM} \\
& \quad \text{while } \|\Delta q\| > \varepsilon \\
& \quad \quad 3) \quad \{p, Q\} \leftarrow DK(q, \text{DH-Workcell}) \\
& \quad \quad 4) \quad \Delta Q \leftarrow Q^T \cdot Q_d \\
& \quad \quad 5) \quad \Delta p \leftarrow p_d - p \\
& \quad \quad 6) \quad \Delta t \leftarrow \begin{bmatrix} Q \cdot \text{vect}(\Delta Q) \\ \Delta p \end{bmatrix} \tag{6.2} \\
& \quad \quad 7) \quad J_{g \text{ Workcell}} \leftarrow DK(q, \text{DH-Workcell}) \\
& \quad \quad 8) \quad \text{Determination of } k_F \\
& \quad \quad 9) \quad \Delta q \leftarrow \text{RRS} \\
& \quad \quad 10) \quad q \leftarrow q + \Delta q \\
& \quad \text{endwhile} \\
& \text{endfor}
\end{aligned}$$

where the sub-index *Workcell* refers to the kinematic chain of the complete kinematic chain of the workcell, i.e. including the linear track and the rotary table.

To determine the convenience of a given posture, the condition number of the Jacobian (with the Frobenius norm, i.e. k_F) was introduced in Section 2.3.2. Moreover, this calculus is going to be done for the isolated 6R KR15/2 manipulator, that is, leaving aside the external joints as justified in 2.4.6. Therefore, the 8th step can be detailed as:

$$\begin{aligned}
& 8) \text{ Determination of } k_F \\
& 8.1) \quad q_{6R} \leftarrow \{0, \theta_2, \dots, \theta_6\} \\
& 8.2) \quad J_{g \ 6R}(q) \leftarrow \text{DK}(q_{6R}, \text{DH-KR15/2}) \\
& 8.3) \quad H_{6R} \xleftarrow{L \text{ (Section 2.4.6)}} J_{g \ 6R} \\
& 8.4) \quad k_F \xleftarrow{\text{eq. (2.41)}} H_{6R}
\end{aligned} \tag{6.3}$$

Algorithm (6.2) is customized in the 9th step for each of the two RRS studied, as it will be described in next sub-sections. It is remarkable that the DH representation of the manipulators depends on the RRS selected, see Section 5.3.3. At this point, it is noteworthy the great influence on the result of the performance vector, h .

ii) Performance criterion vector, h

As explained in Section 5.3.1, the manipulator is required to track successive target positions as *primary task*, but in addition one can try to achieve secondary goals by suitably choosing h . It could be considered as having a *virtual force* which attempts to push the configuration of the manipulator away from a critical area in the configuration space [9].

Nevertheless, as explained in Chapter 5, it is important to remark the different signification of h in both VJM and TDM. In case of the VJM, fully based on eq. (5.20), h is a motion projected on $\mathcal{N}(J)$ (i.e., the tool tip is not moved by the action of h), but in case of the TDM this secondary motion may not be constant in the base frame (i.e., from the base frame, a movement can be appreciated through the symmetry axis of the tool tip). It was depicted in Figure 5.11.

- **Joint-limit avoidance**

When considering the performance vector (h) for joint-limits avoidance, the Yoshikawa's formulation (5.24) is widely used [8]. In this case the performance criterion can be written as to maintain the manipulator as close as possible to the *mid-joint posture*, i.e. as far as possible from its *mechanical* joint limits, namely

$$p_{jnt} = \frac{1}{2}(q - q^{\text{mid}})^T W_{jnt}(q - q^{\text{mid}}), \quad \text{with } q^{\text{mid}} = \frac{q^{\text{min}} + q^{\text{max}}}{2} \tag{6.4}$$

It is remarkable that the *mid-joint posture* of the KR 15/2 is such a non-functional posture that it is not appropriate as a reference (Figure 6.9, right). Also the *best conditioned posture* could be considered as reference (in fact, it seems to be more logical and desirable) but, as shown in Figure 6.9 (left), it is quite near of some mechanical joint limits. Thus, and for the scope of this thesis, the commonly used HOME posture depicted in Figure 6.8 will be taken as the reference posture ($q^{\text{ref}} \equiv q_0$) for eq. (6.4) as a compromise between both objectives.

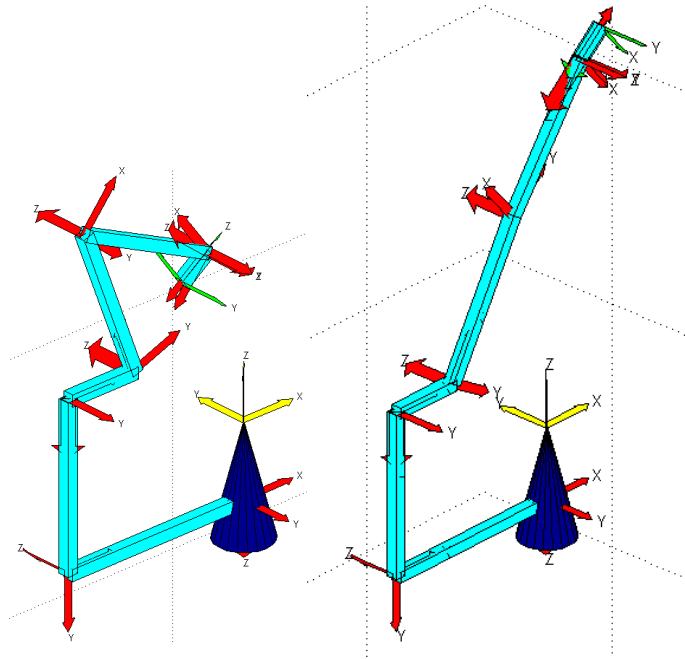


Figure 6.9. Left, best conditioned posture for the 6R KR 15/2 manipulator deduced in Section 2.4.6. Right, mechanical mid-joint posture.

The setting of the weighting diagonal matrix W_{jnt} of equation (6.4) is very important for the success, as demonstrated below.

- **Best conditioning (k_F)**

In addition, the k_F -condition number is also taken into account to achieve a definitive h . Thus, for the scope of this thesis, the performance criterion (5.26) can be rewritten by using the k_F -condition number as

$$p_{cond} = \frac{k_F}{2} (q - q_{Ts})^T W_{cond} (q - q_{Ts}) \quad (6.5)$$

This *performance criterion* is activated when the k_F -condition number passes over a preset threshold value, ζ . At this instant, the corresponding configuration, q_{Ts} , is recorded. Thus, the algorithm considered takes the form:

$$\begin{aligned} & \text{if } \frac{1}{k_F} < \zeta \\ & \quad \text{if } q_{Ts} = \emptyset \\ & \quad \quad q_{Ts} \leftarrow q_{actual} \\ & \quad \text{end} \\ & \quad h \leftarrow -\nabla p_{cond} = -W_{cond} \cdot k_F \cdot (q - q_{Ts}) \\ & \text{else} \\ & \quad q_{Ts} \leftarrow \emptyset \\ & \quad h \leftarrow \emptyset \\ & \text{end} \end{aligned} \quad (6.6)$$

Compared to k_F , the inverse of k_F has the advantage of being comprised between 0 and 1 (best conditioned). It also makes easier the graphical representation and comparison in further sections.

Again, in (6.5) the choice of the *weight*, W_{cond} , is a major difficulty to implement due to the subjectivity.

- **Combined performance criterion**

Finally, the two secondary tasks described above, joint-limits and kinematic singularity avoidance can be combined into a unique performance criterion vector, which is to maintain the manipulator as close as possible to the reference posture (HOME) and as far as possible of bad conditioned postures at the same time. The objective function could be written as:

$$P = P_{jnt} + P_{cond} \quad (6.7)$$

By tuning both W_{jnt} and W_{cond} , the relative importance between the two sub-tasks is adjusted. Vector h is thus chosen as the gradient of p , namely:

$$h = -\nabla p = -\nabla(p_{jnt} + p_{cond}) = h_{jnt} + h_{cond} \quad (6.8)$$

$$h_{jnt} = -W_{jnt}(q - q^{ref}) \quad (6.9)$$

$$h_{cond} = -W_{cond} \cdot k_F \cdot (q - q_{Ts}) \quad (6.10)$$

Choosing both W can be critical for the performance of the RRS and traditionally this task has been set based on trial and error. This matter will be considered when performing the numerical tests.

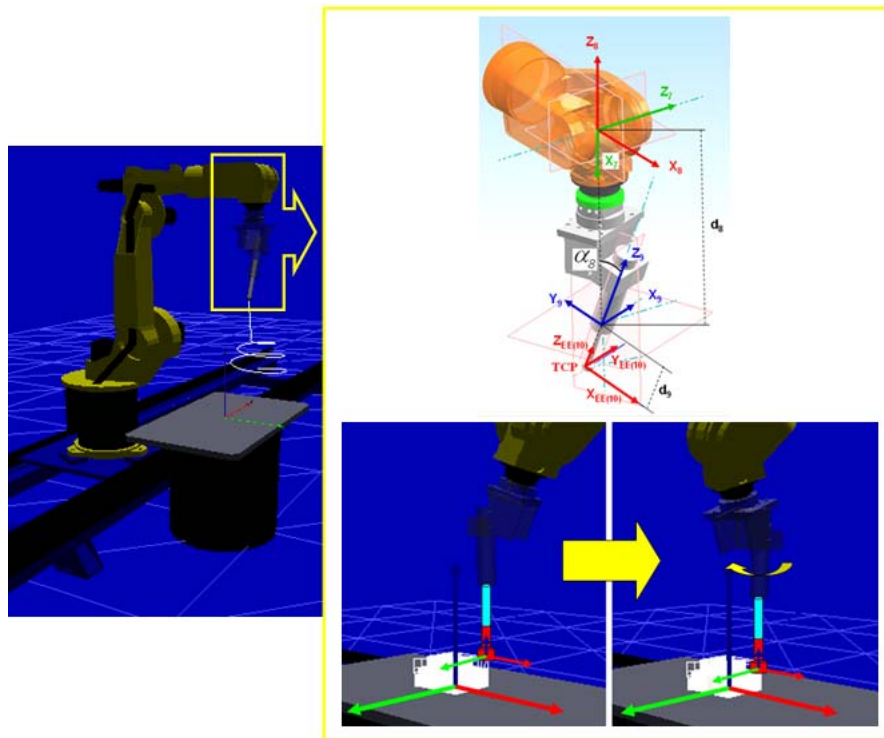


Figure 6.10. Additional virtual joint, associated with a rotation in Z_9 .

iii) Algorithm for the VJM

The VJM was profusely described in Section 5.3.2.i). In fact, the most significant implication is the consideration of an additional rotary joint in the tool tip around the tool symmetry axis. Therefore, the DH

model for the IDF's workcell has an additional line more than expected in the real model (namely, the virtual joint). It was detailed at Section 5.3.3 (see Figure 6.10).

The RRS in the 9th step of the algorithm (6.2) is programmed with Matlab following the scheme of Section 5.3.1.v), by using Householder reflections. Again, it is justified by the fact that these computations are performed with finite precision. Hence, in order to keep round-off errors as low as possible, this algorithm avoids the direct calculation of the generalized inverse of the Jacobian matrix. The Matlab algorithm can be resumed as follows [7]:

$$\begin{aligned}
 &1) J^T \leftarrow J' \quad (' \text{ designates the transpose of a matrix in Matlab}) \\
 &2) [Q \ R] = qr(J^T) \quad (\text{Matlab's function } qr \text{ makes the} \\
 &\quad \quad \quad \text{orthogonal-triangular } QR \text{-decomposition}) \\
 &\quad 2.1) H \leftarrow Q' \\
 &\quad 2.2) U \leftarrow R(1:6,:) \\
 &3) r \leftarrow \Delta t - J \cdot h \quad (6.11) \\
 &4) y_1 \leftarrow \text{forward}(U', r) \quad (U^T \cdot y_1 = r, \text{ solved by forward substitution,} \\
 &\quad \quad \quad \text{with U being a upper triangular matrix}) \\
 &5) y \leftarrow [y_1; \text{zeros}(3,1)] \\
 &6) k \leftarrow H' \cdot y \\
 &7) \Delta q = k + h
 \end{aligned}$$

iv) Algorithm for the TDM

The TDM was also described in Section 5.3.2.ii) and complemented in Section 5.3.3 with the projection on $\mathcal{N}(J)$. In this case, the DH model is the one described in Chapter 2, but with the adequate attached length of the final link until the tool tip. Thus, the position and orientation of the tool tip regarding the tool-holder is known by means of the corresponding homogeneous matrix (${}^9T_{TCP}$), as detailed at Section 5.3.3 and Figure 6.11.

Again, it is justified the use of the Householder reflections to solve (5.65), but with some modifications as shown in the following algorithm:

- 1) $J^T \leftarrow J'$ (' designates the transpose of a matrix in Matlab)
- 2) $[Q \ R] = qr(J^T)$ (Matlab's function qr makes the orthogonal-triangular QR -decomposition)
 - 2.1) $H \leftarrow Q'$
 - 2.2) $U \leftarrow R(1:6,:)$
- 3) $\Delta t^* = T \cdot \Delta t + \begin{bmatrix} e \cdot e^T \cdot A \cdot h_1 \\ 0_{3 \times 1} \end{bmatrix}$
- 4) $r \leftarrow \Delta t^* - J \cdot h_2$
- 5) $y_1 \leftarrow \text{forward}(U', r)$ ($U^T \cdot y_1 = r$, solved by forward substitution, with U being an upper triangular matrix)
- 6) $y \leftarrow [y_1; \text{zeros}(2,1)]$
- 7) $k \leftarrow H' \cdot y$
- 8) $\Delta q = k + h_2$

(6.12)

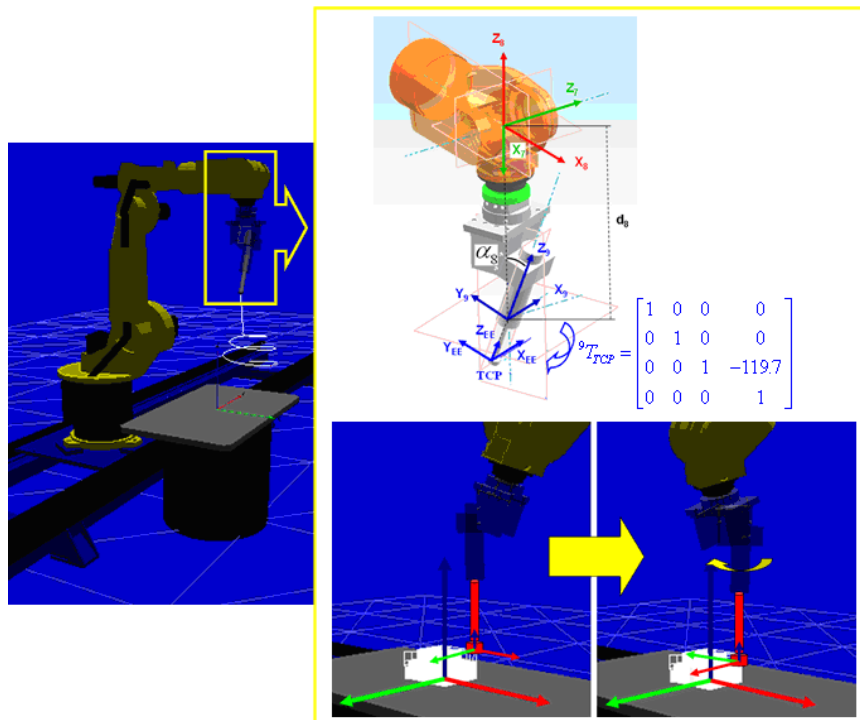


Figure 6.11. Representation of the EE in the TDM test. The transformation matrix towards the tool tip is expressed as a displacement on Z_0 in mm.

6.3.3. Analysis and results.

The exposed algorithms were run in Matlab for the test described previously. At the same time, the workcell was sketched and virtually animated with the aid of the Matlab's Toolbox *Hemero* [10].

For each possible treatment, namely:

- (a) the VJM,
- (b) the TDM,
- (c) the TDM combined with a projection on $\aleph(J)$, eq. (5.65);

the joint values and the inverse of k_F were recorded. Moreover, in the case (c), it may be possible to highlight the convenience of evaluating as h_1 and h_2 (at (5.65)) the respective values of h_{jnt} and h_{cond} (of (6.8)).

For a value of $\zeta = 0.5$ in algorithm (6.6), the following tests were performed:

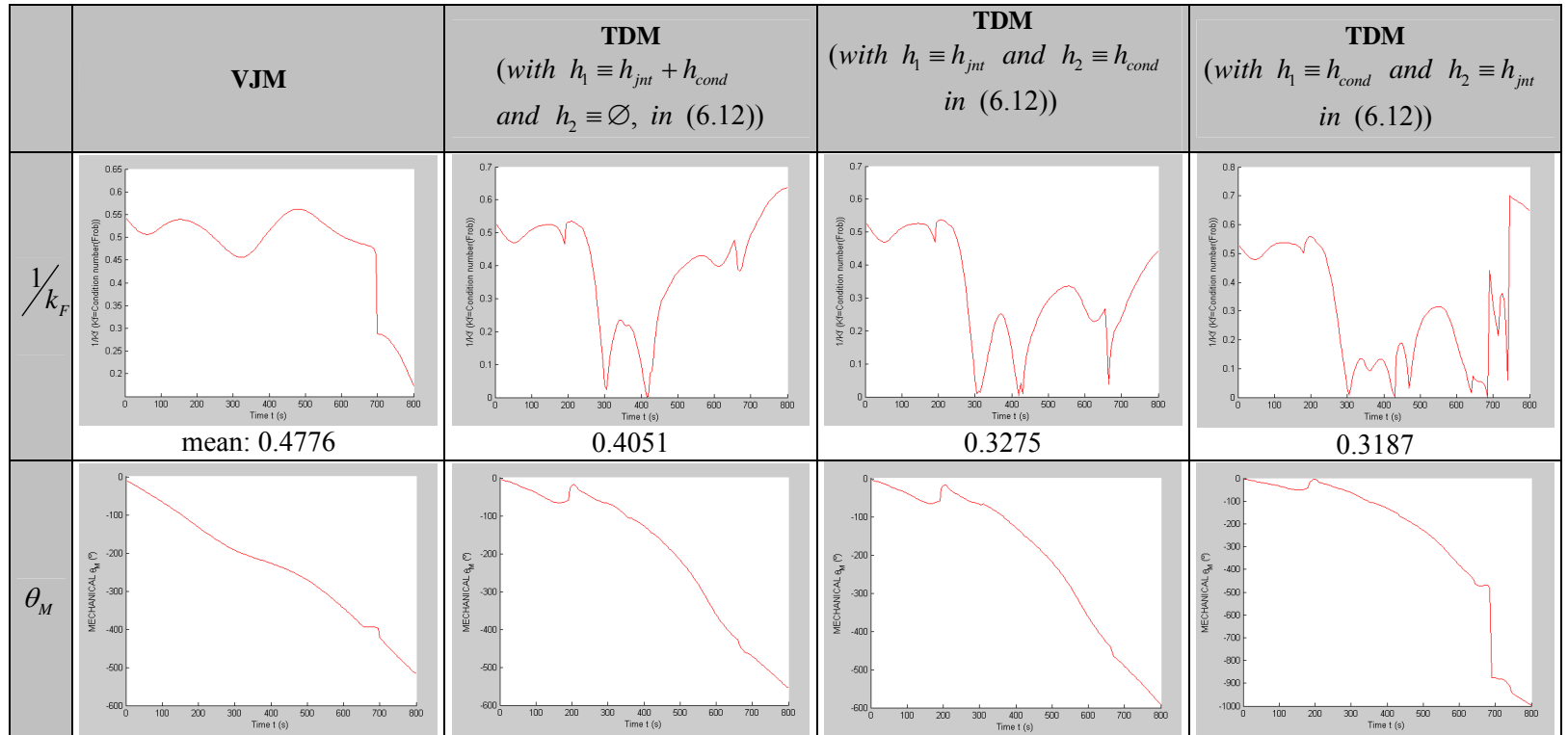
i) Constant weighting vector for the combined performance criterion.

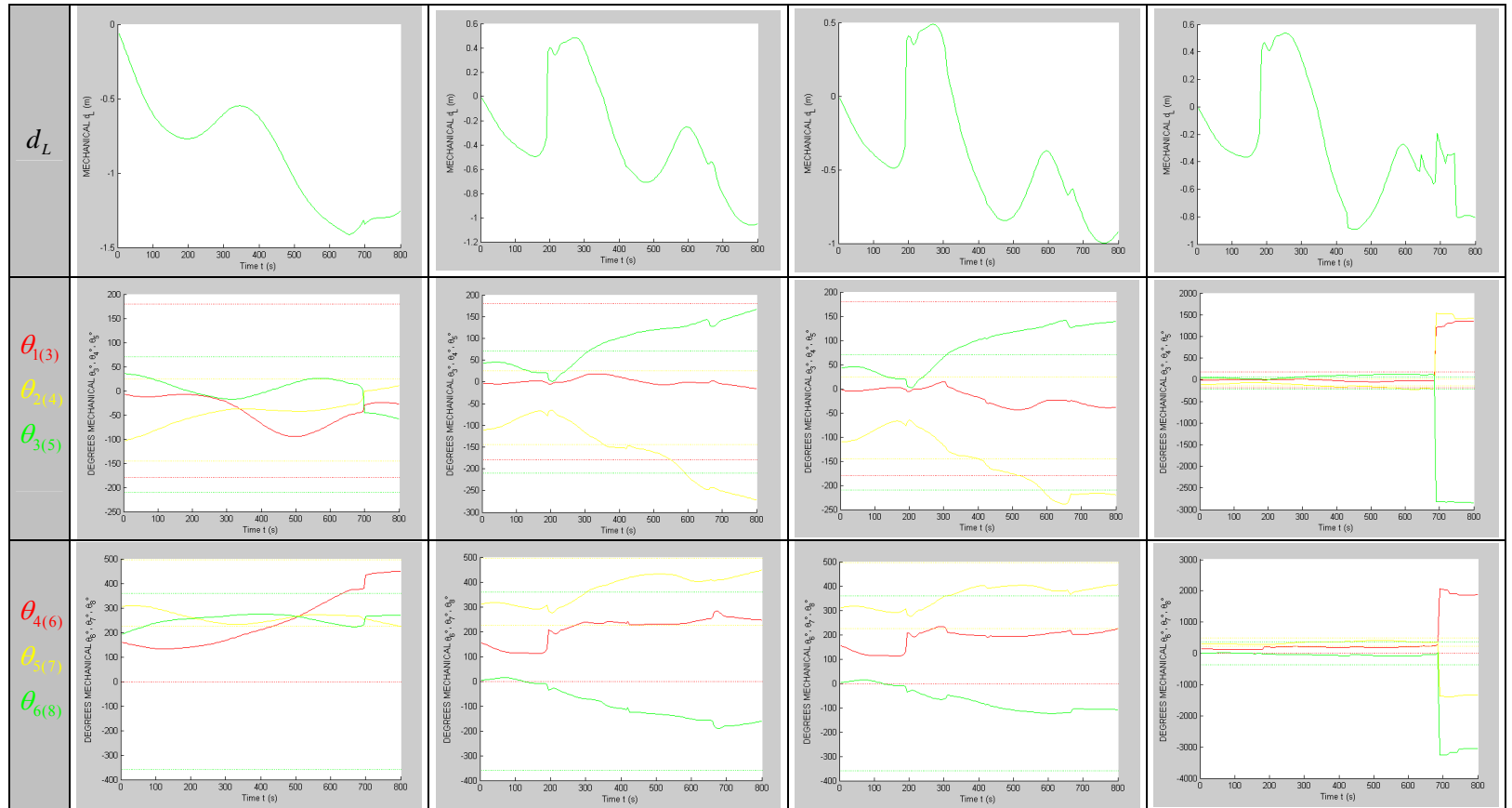
In order to perform the first attempt of evaluation of the VJM and TDM methods, two constant diagonal weighting matrixes are assumed [8], namely

	θ_M	d_L	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	$\theta_{7(VJM)}$
W_{jnt}	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
W_{cond}	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

Table 6.4. From left (element (1,1)) to right (element (8,8)), diagonal weighting matrixes for the combined performance criterion.

The results and comparison amongst the cases studied are graphically depicted in the Table 6.5:





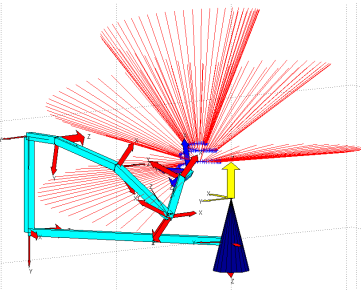
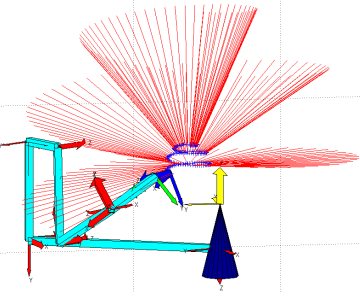
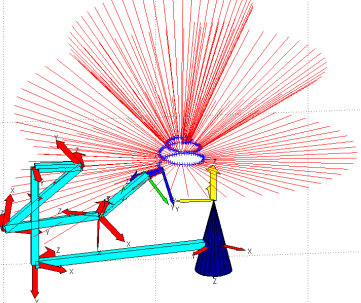
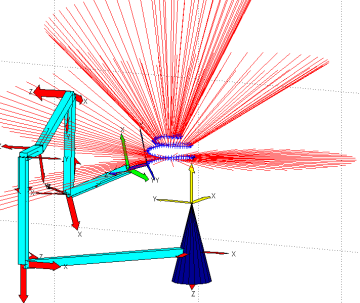
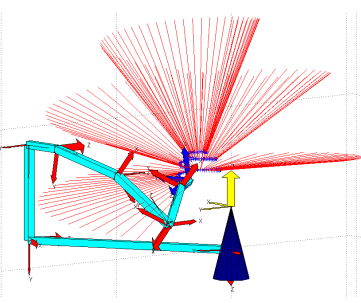
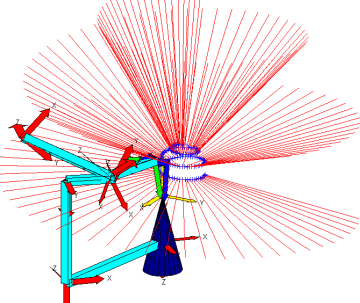
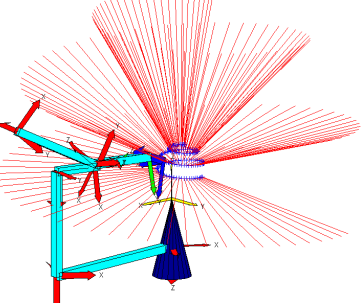
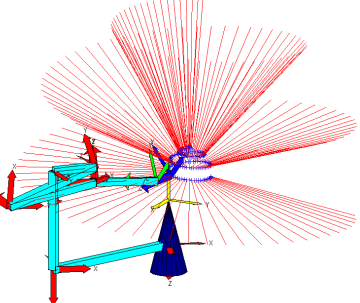
Final posture				
Worst k_F posture				

Table 6.5. Experimental results for the simulation in the studied workcell of the VJM and TDM algorithms. In the TDM, two variations including a projection in $\mathcal{N}(J)$ are studied, according to case (c).

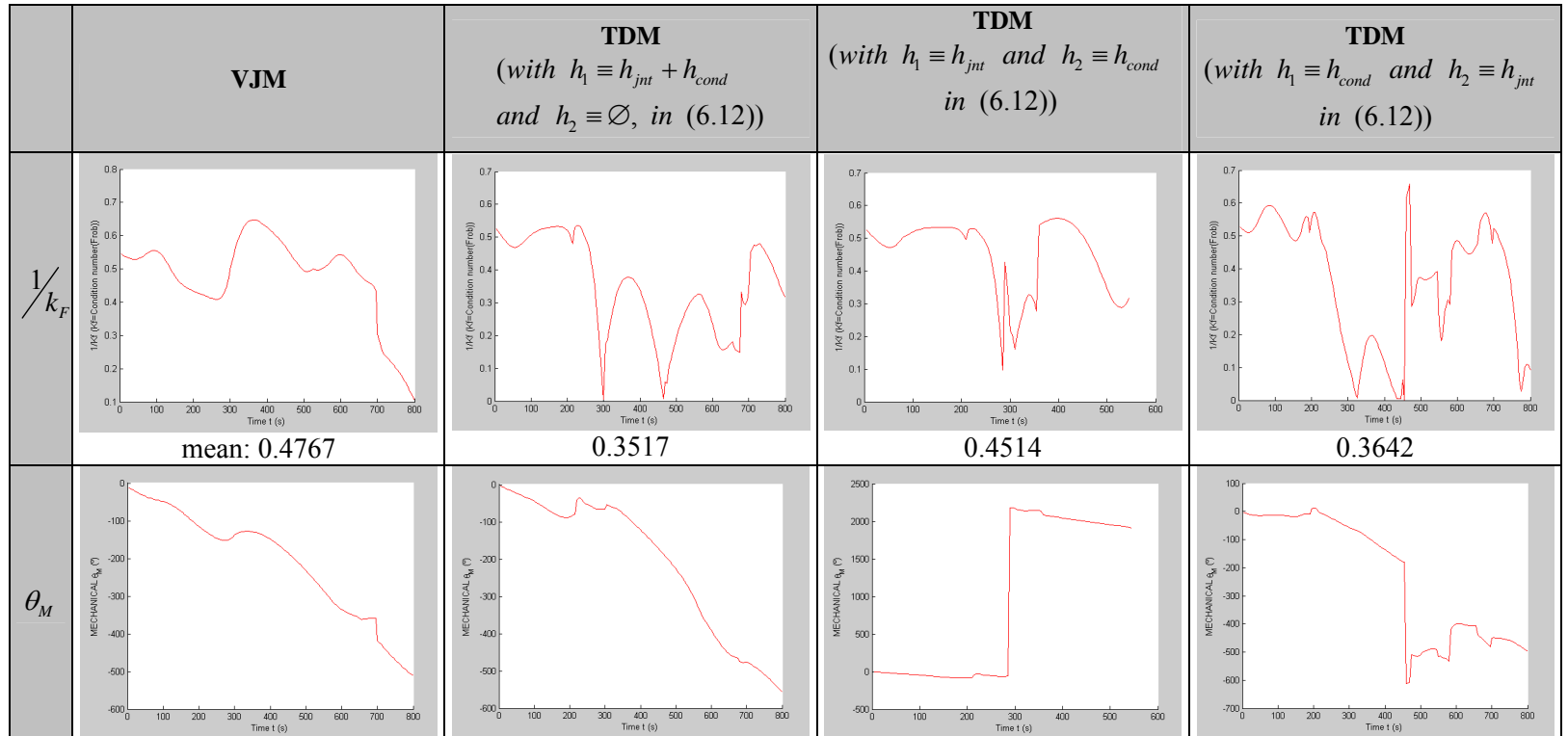
From the analysis above, the VJM seems to be more robust configuring the consecutive postures along the path and taking into account the criterion of the proximity to the reference posture and best k_F .

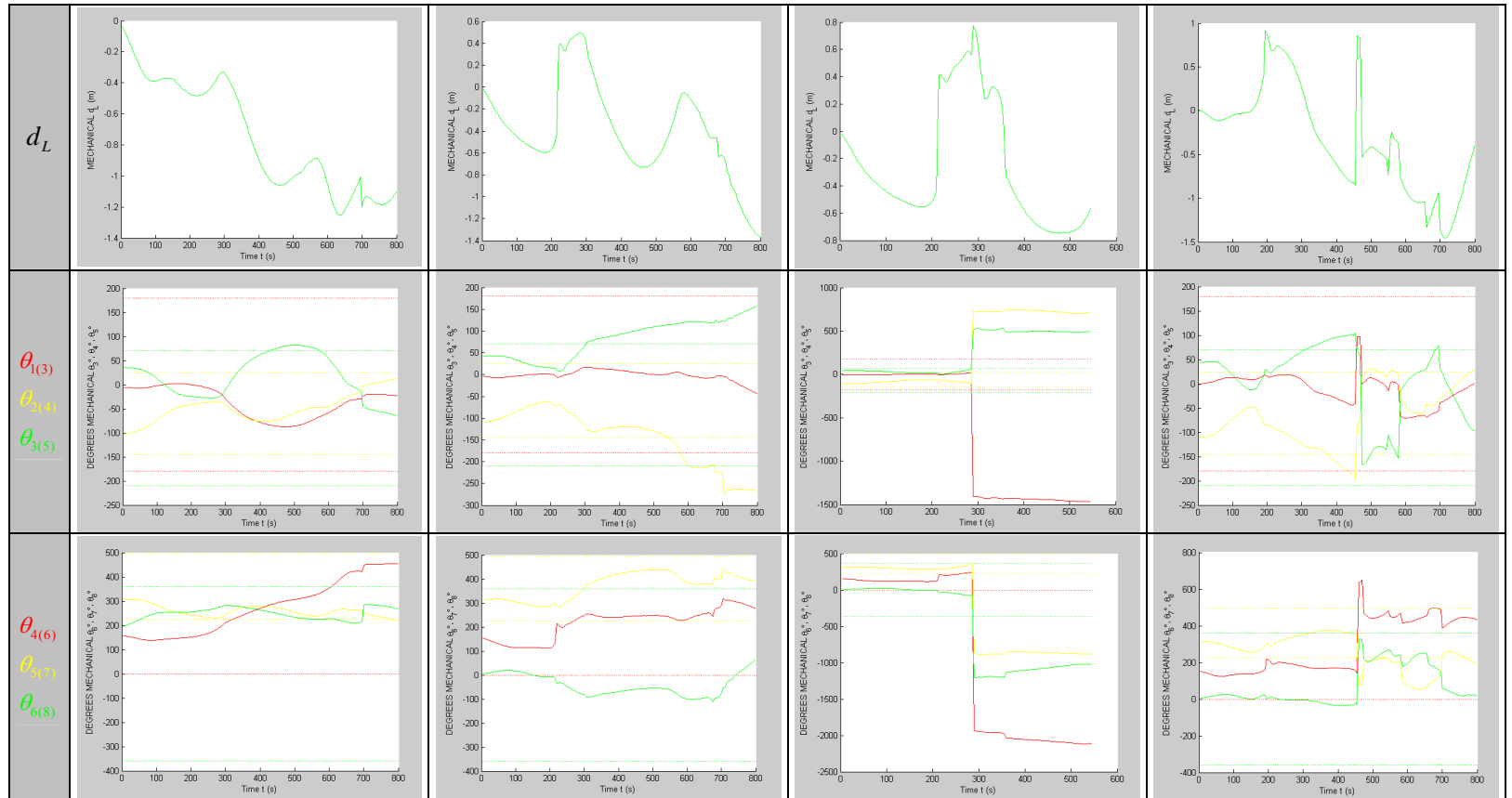
A second attempt can be done varying the magnitude of the weights. It is noteworthy that the more weight, the faster reaction can be expected in the manipulator. Thus, two constant diagonal weighting matrixes are assumed, namely, with a 10-times bigger order of magnitude:

	θ_M	d_L	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	$\theta_{7(VJM)}$
W_{jnt}	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
W_{cond}	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Table 6.6. From left (element (1,1)) to right (element (8,8)), diagonal weighting matrixes for the combined performance criterion.

The results and comparison amongst the cases studied are graphically depicted in the Table 6.7:





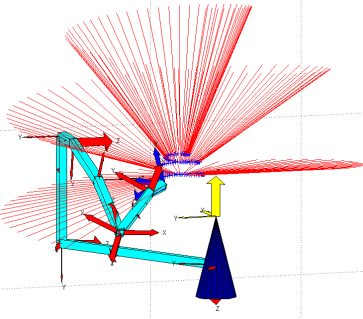
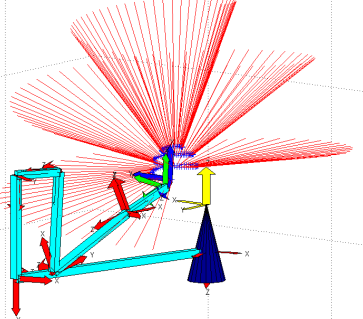
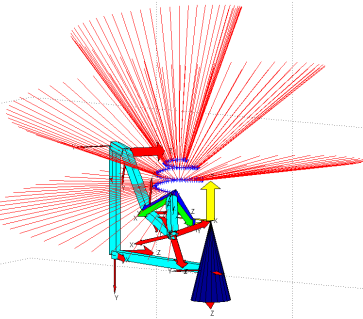
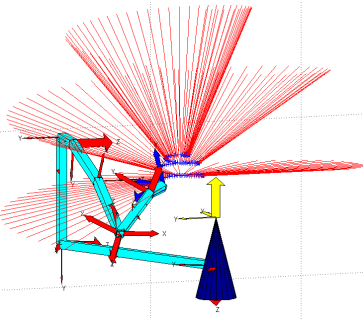
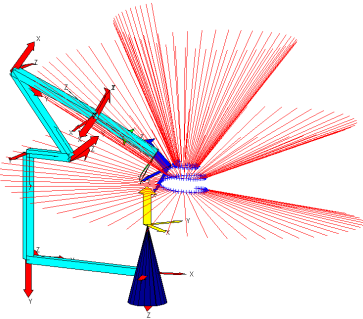
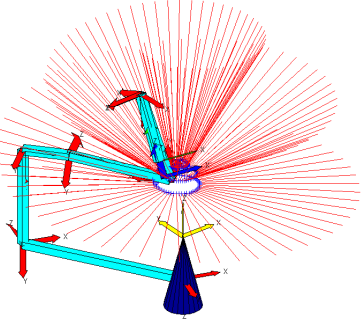
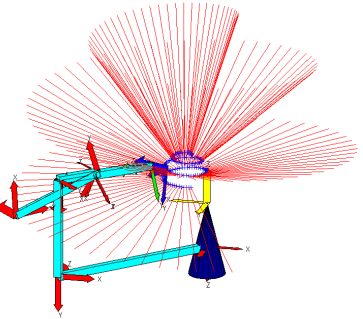
Final posture			\emptyset (none)	
Worst k_F posture				

Table 6.7. Experimental results for the simulation in the studied workcell of the VJM and TDM algorithms, with the constant weights of Table 6.6. In the TDM, two variations including a projection in $\mathfrak{N}(J)$ are studied, according to case (c).

The VJM shows the more robust behaviour, and almost similar to the previous study. Nevertheless, θ_3 goes out of a mechanical limits so the first attempt of VJM can be regarded as most convenient. Again, the TDM shows a more unstable and even unpredictable behaviour, much sensible to the weight performance.

ii) Adapted Fuzzy weighting vector for the combined performance criterion.

As recommended by several authors [8][11][12][13], higher weights are assigned to those joints that are supposed to be more reactive when lowering the condition number or being far of the reference posture. Those studies assigned the weights depending on the significance of the joints and according to an expert knowledge. Nevertheless, in case of milling tasks where the tool *pose* (and hence the robot *posture*) changes constantly, it seems to be desirable to identify an appropriate value for the weights at each configuration and in a reasonable time.

In practice, the implementation reported for the configuration of a fuzzy engine controlling the position (Section 6.2.) gives the key to configure a similar one but performing, on the basis of a rule base, the importance (namely, the *weight*) to be associated to each joint.

The steps to develop such a fuzzy controller with the Matlab's *Fuzzy* Toolbox were described in Section 6.2.1. , and the interactivity with this controller from the postprocessor module was shown more explicitly in sub-section *vii*). Therefore, the description to be done in this section points to the variable and knowledge base definitions.

- **Variable definition**

The condition number is expected to be decrease when the robot acquires a posture near the extended arm or the wrist singularities (described in Chapter 2). In this case, joints θ_3 and θ_5 had a direct implication, and also the additional joints (θ_M, d_L) in order to avoid this posture.

In the case of the maintenance of a reference posture, where all joints are implied, different weights are assigned to the joints articulating the *gross* and *fine* positioning described in Chapter 2, as well as the

additional joints. In fact, it can be convenient to work near a reference posture of the joints doing the gross positioning while a fine orientation is being done, so it seems to be logical making different assignments.

Based this reasoning, the output variables of the fuzzy system are those weights associated to the joints which are more related to a critical change in both aspects described (from Table 6.3):

	θ_M	d_L	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	$\theta_7(VJM)$
W_{jnt}	w_{Mjnt}	w_{Ljnt}	0.01	0.01	w_{3jnt}	0.01	w_{5jnt}	0.01	0.01
W_{cond}	w_{Mcond}	w_{Lcond}	w_{gross}	w_{gross}	w_{gross}	w_{fine}	w_{fine}	w_{fine}	0.01

Table 6.8. From left (element (1,1)) to right (element (8,8)), diagonal weighting matrixes for the combined performance criterion. The fuzzyfied weights are assigned to the most significant joints according to experience.

On the basis of those data and after studying which robot joints affect the most the conditioning and the risk of joint limit reaching, the input variables are structured. Two input variables are defined in case of the conditioning (θ_3, θ_5 , as justified in Section 6.2.1.) and three in case of the maintenance of the reference posture ($\theta_2, \theta_3, \theta_5$), (Figure 6.12, in yellow). Moreover, it is noteworthy that the absolute value of θ_5 , namely $|\theta_5|$, will be considered due to symmetry in the range of this joint.

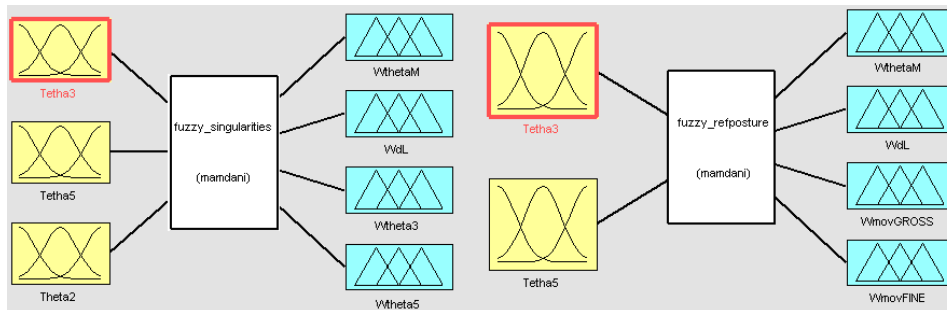


Figure 6.12. Left, graphical representation of the fuzzy engine determining the weights for the reference posture maintenance criterion. Right, the fuzzy engine giving the weights for the singularity avoidance criterion.

- **Clusterization of input and output spaces**

In each input and output spaces, the number of clusters is related to the linguistic etiquettes assigned, according to the experience. In case of the input spaces (θ_2, θ_3 , or θ_5), they are divided in three triangular clusters (Figure 6.13). It can be noticed that the functions neither are equidistant nor have identical form. It depends on the expected reactions.

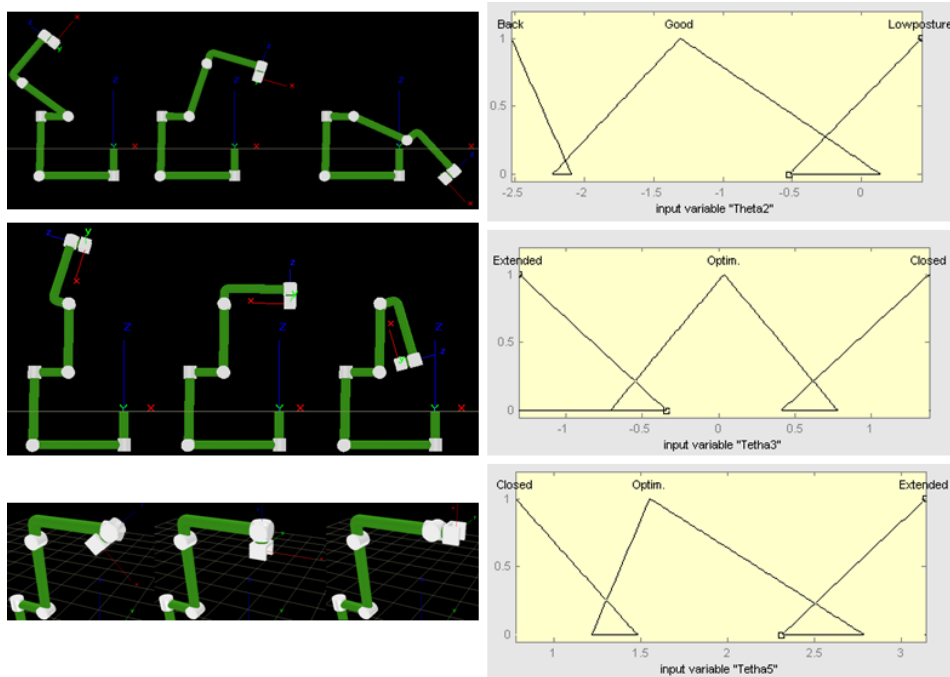


Figure 6.13. From left to right, representation of the peak posture of each of the three clusters in which the input spaces are divided.

The output spaces are different for each fuzzy inference system, depending also on the experience. In each case, the weights are comprised between 0 and 0.05 (0.025 for the additional joints when considering the reference posture maintenance).

Moreover, in practice the additional joints are considered after the most adequate solutions involving the main chain of the manipulator. As a

result, only two clusters are considered for the external joints (one giving a very low weight, almost 0) while three clusters are considered in the case of the joints of the main chain (θ_1 to θ_6), the one on the left giving a major weight if required (Figure 6.14).

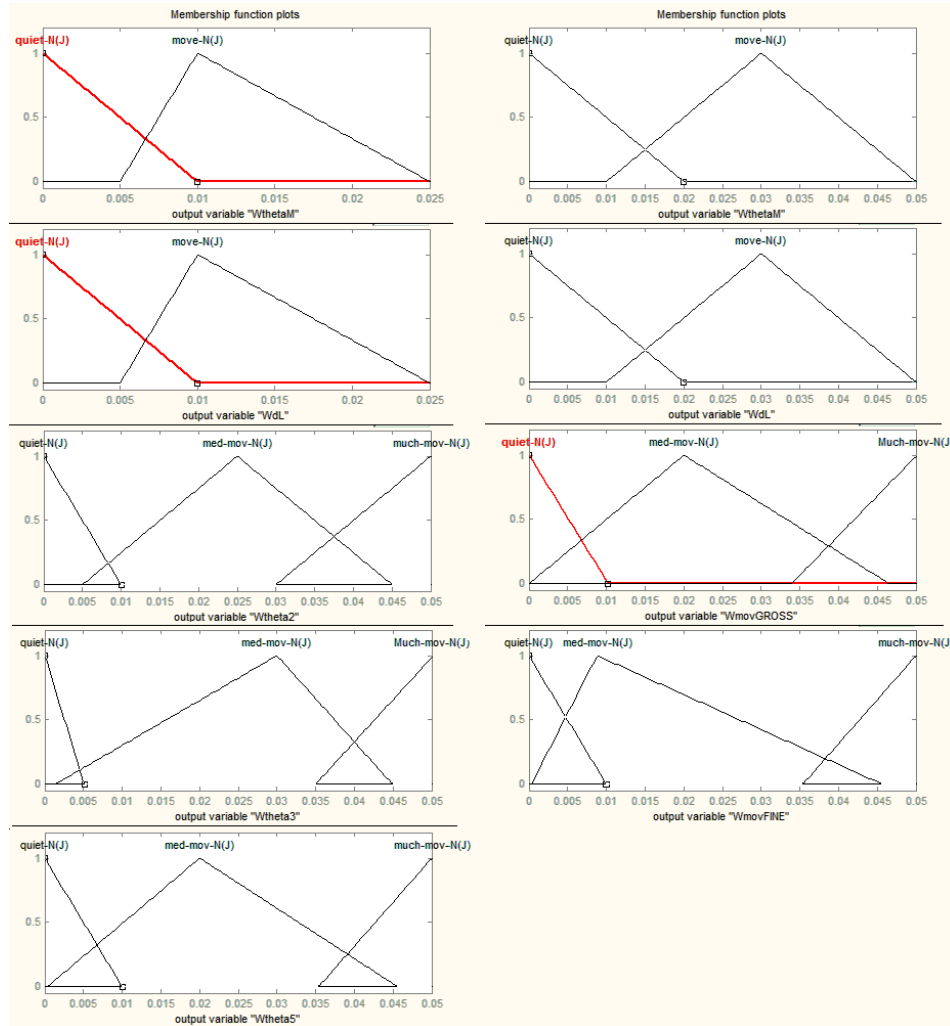


Figure 6.14. Output spaces for the weight assignment: left, for the reference posture criterion; right, for the singularity avoidance criterion.

- **Knowledge base**

The last requirement to run the inference engine consists in the rule base relating the input and output spaces. Those “*if-then*” rules comprise a number of 4-5 up to a maximum of 12. Teorically, giving more rules can be cumbersome and comes up with the desired simplicity of a fuzzy inference system.

In case of this trial, some basic rules are taken into account, as shown in Figure 6.15. Next, with those simple rules, the results obtained are shown.

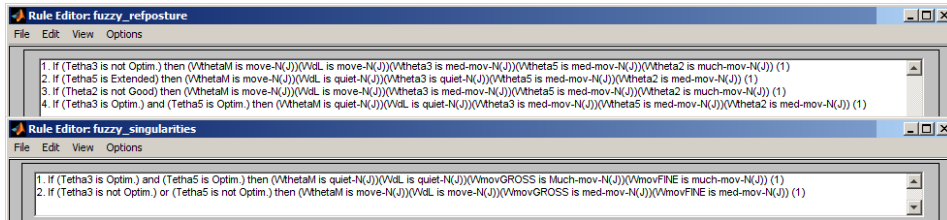
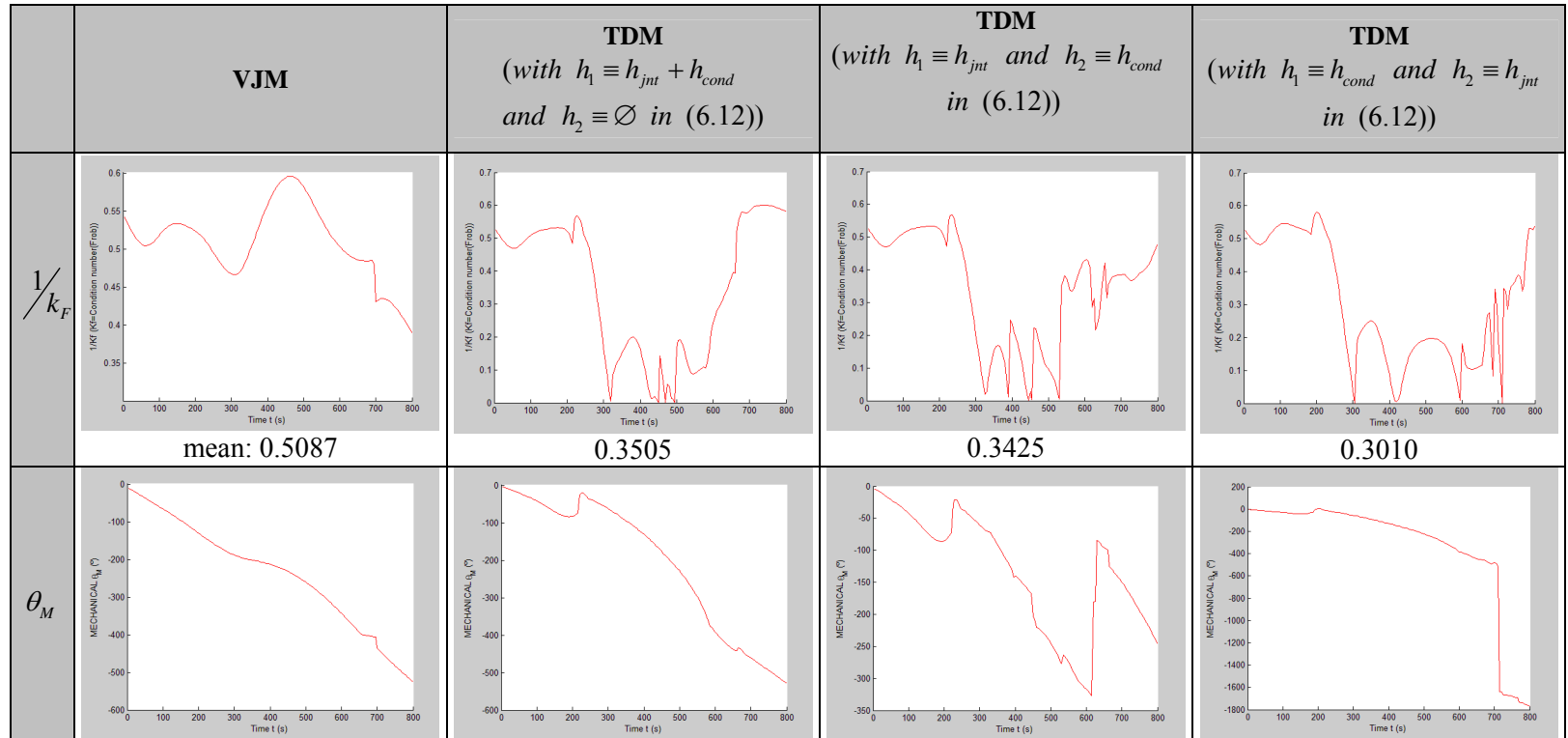


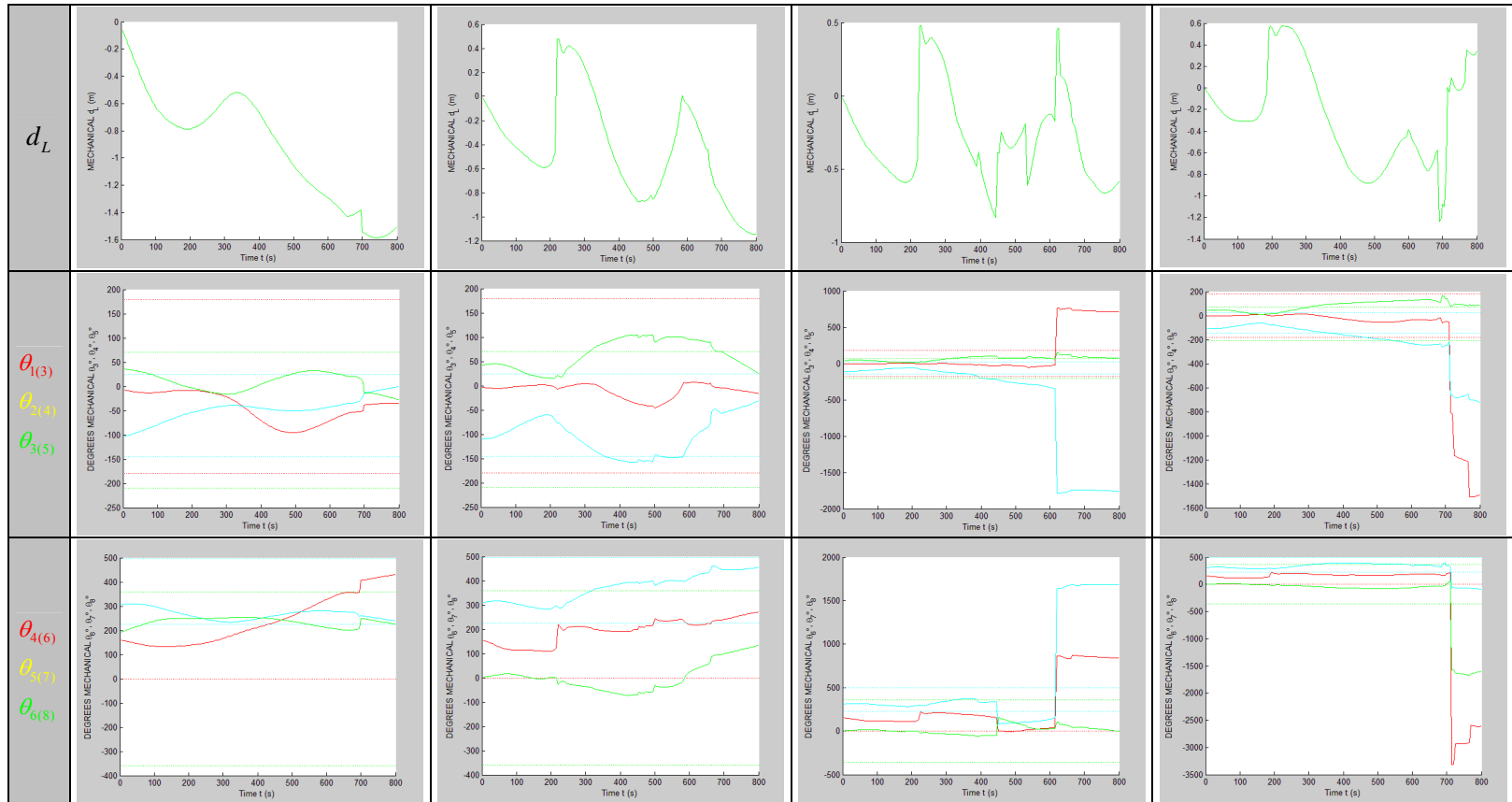
Figure 6.15. MATLAB’s Fuzzy Toolbox has a Rule Editor to easily manage the *if-then* rules relating the *input* and *output* spaces. Four rules were created for the reference posture criterion (up), and two for the singularity avoidance criterion (down).

- **Results with the adapted fuzzy weighting vector**

Due to the fact that previous studies showed a better performance of the VJM, the implementation of the adapted fuzzy weighting vector was done only for this method.

With the suppositions of the preliminary sections, the results and comparison amongst the cases studied are graphically depicted in the Table 6.9:





Final posture				
Worst k_F posture				

Table 6.9. Experimental results for the simulation in the studied workcell of the VJM and TDM algorithms, with the adapted weights of Table 6.8 via fuzzy inference. In the TDM, two variations including a projection in $\mathfrak{N}(J)$ are studied, according to case (c).

From the analysis above, it can be observed that the VJM again offers the best k_F average, but also the more robust behaviour. In this trial, the TDM has a quite unstable behaviour again, although resulting in a better conditioned final posture compared to VJM. The worst conditioned point in the TDM is achieved due to wrist singularities, as depicted in Table 6.9.

Following attempts are going to be done with the VJM, taking profit from its robustness but considering a periodic IKP revision to perform a better control on the final k_F .

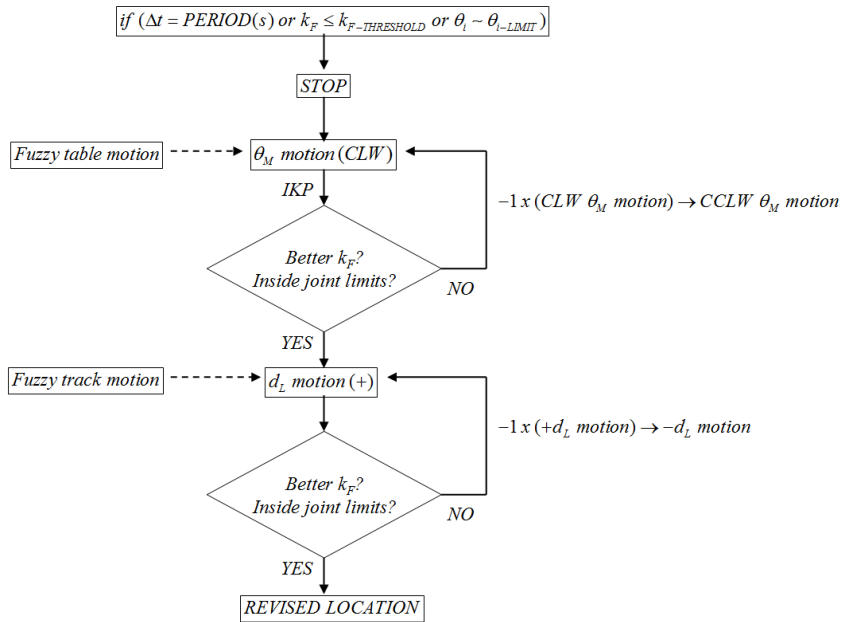


Figure 6.16. Proposed Fuzzy revision for the studied workcell.

6.3.4. Periodic revision by fuzzified IK analysis.

Common milling tasks are compound of a succession of toolpath trackings allowing a posture revision between them. In previous trials, a continuous toolpath has been considered showing the robustness of VJM when compared to TDM.

However, to practical effects, it can be convenient to take profit from a periodic revision of the posture. It can be done at a set of points on the toolpath by means of the IK of position, as shown in Section 6.2. For this, the rotary table and the track are moved at those points to try to radically improve the k_F , taking

into account the desired pose of the TCP, the actual position on this additional joints and the resulting IKP analysis.

As shown in Figure 6.16, different criteria can be considered for this periodic revision, namely the time from the previous revision, a k_f threshold or reaching any of the joint limits in the kinematic chain of the manipulator. In all those cases, the position of the table can be first revised to get better posture. This is due to the fact that with this, the major improvement is achieved without moving track. Thus, in a certain manner, it is also an observation done from experience attending to precision of the manipulator (as noted in Chapter 3). After that, another fuzzified track motion can be considered, which for this case will be smaller than on the contrary.

With this supposition and with the more robust of the previous methods tested, the VJM, the results obtained are shown in the following table:

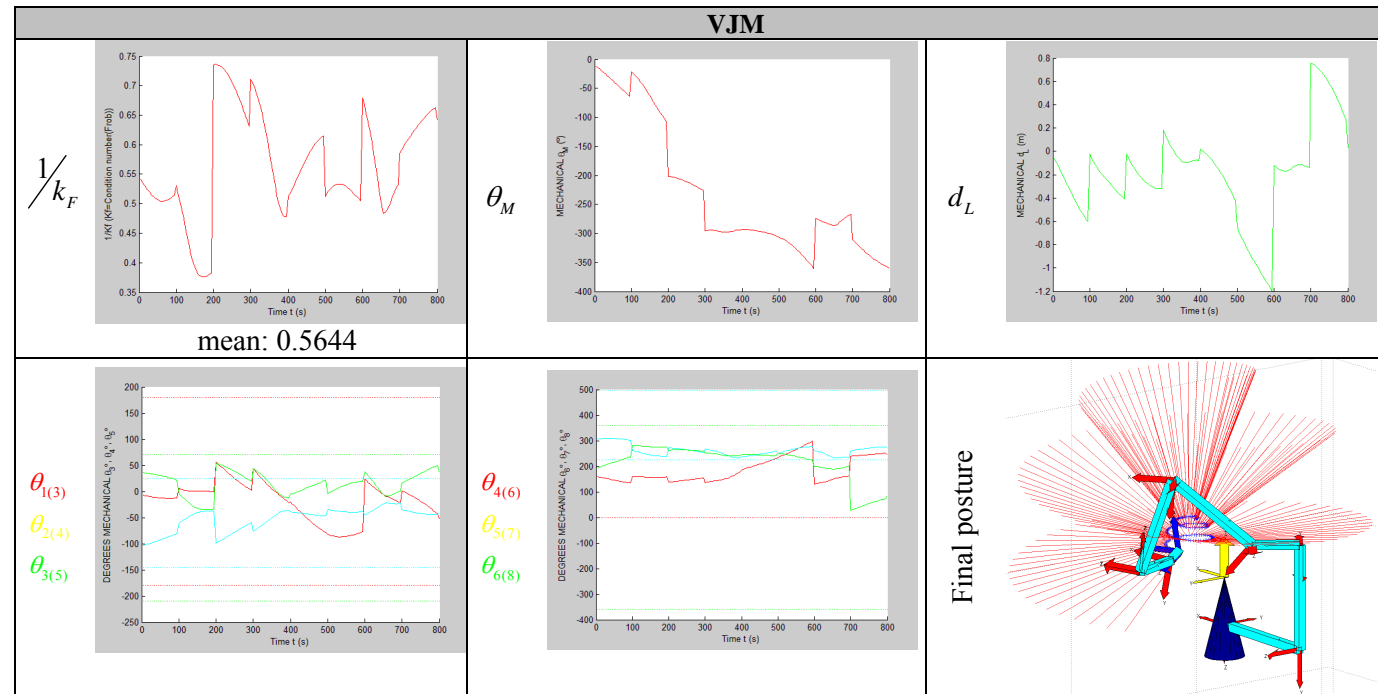


Table 6.10. Experimental results for the simulation in the studied workcell of the VJM with the implementation of the adapted fuzzy weighting vector and a periodic IKP position analysis as depicted in Figure 6.16

6.3.5. Comparison of the previous VJM improvements

Figure 6.17 compares the conditioning achieved with the different VJM trials of the previous sections. It can be appreciated the progressive better conditioning of each of the successive improvements done.

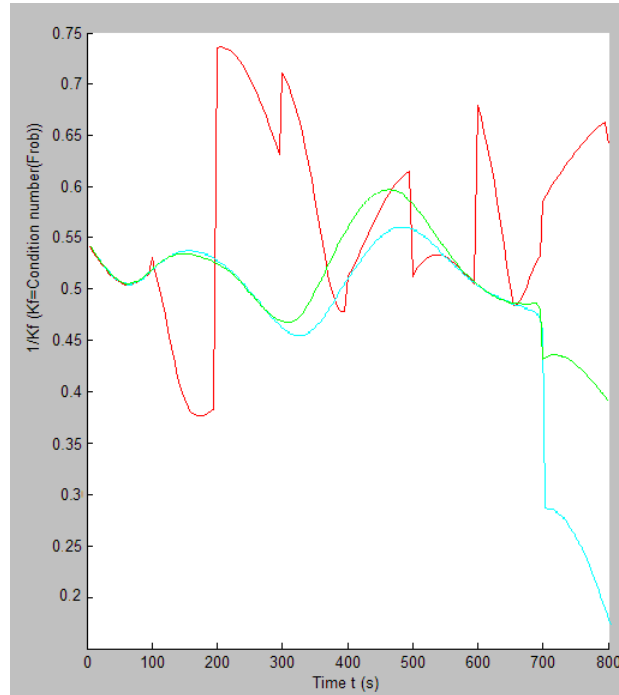


Figure 6.17. Comparison of the conditioning achieved with the different VJM trials: blue, with constant weighting vector ($w=0.01$); green, with fuzzy adapted weighting vector; and red with fuzzy adapted weighting vector and a periodic revision of the IKP.

Moreover, the last method, assisted by the IKP revision, can be considered more robust than the previous implementations. In fact, the worst conditioned posture, at $t=180s$, is about 0.4 like the method only using a fuzzy adapted weighting vector, but in this case this posture is rapidly corrected in the following revision at $t=200s$ to continue with the milling process.

In addition, the implementation of those methods is to be done in the real robot, and the last one guarantees the postures that are most continuously approachable in the real workcell (Figure 6.18).

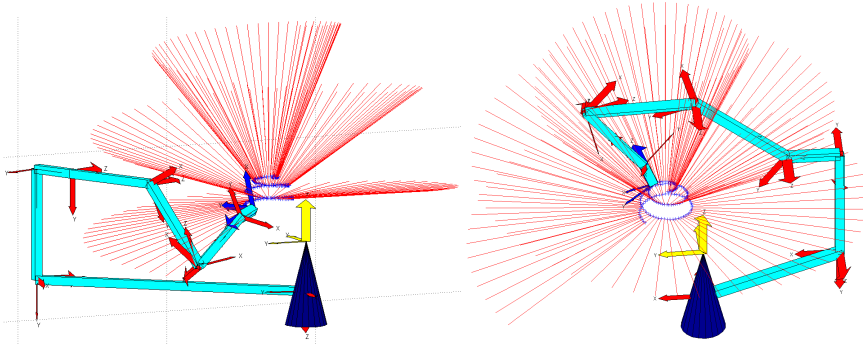


Figure 6.18. While the condition number is almost the same ($k_F=0.4$), the worst posture achieved with the periodically revised method (right) has a better performance for continuous milling purposes.

REFERENCES

- [1] M. D. J. Hayes, M. L. Husty and P. J. Zsombor-Murray; "Singular Configurations of Wrist-Partitioned 6R Serial Robots: a Geometric Perspective for Users", Carleton University (Canada), 2003
- [2] R. Reyero, C. F. Nicolas; "Sistemas de control basados en lógica borrosa : fuzzy control", Omron Electronics, 1995. ISBN 8492032626
- [3] L. Huo; "Inverse kinematics of robotized functionally and intrinsically redundant tasks"; Departement de Genie Mecanique - Ecole Polytechnique de Montreal, 2006.
- [4] A. Ollero; "Robótica: Manipuladores y robots móviles", Marcombo, 2001. ISBN 8426713130
- [5] J. Andres, L. Gracia, J.Tornero; "Inverse kinematics of a redundant manipulator for CAM integration. An industrial perspective of implementation", unpublished (submitted to ICM09)
- [6] J.-S. Roger Jang, N. Gulley; "*Fuzzy Logic Toolbox: User's Guide*"; Revised for Version 2.2.7 (Release 2008a), The MathWorks, Inc. 2008
- [7] <http://www.mathworks.com/access/helpdesk/help/techdoc/ref/> (Matlab Function Reference)
- [8] Huo L. and Baron L.; "The joint-limits and singularity avoidance in robotic welding"; *Industrial Robot: An International Journal* 35/5 pp 456–464 (2008)
- [9] Kemeny Z.; " Mapping, detection and handling of singularities for kinematically redundant serial manipulators "; *Periodica Polytechnica Electr. Eng.*, 46 (2002), pp 29-45.
- [10] J. I. Maza, A. Ollero; "Hemero: Herramienta MATLAB/Simulink para el estudio de manipuladores y robots móviles", Marcombo, 2001.
- [11] Huo L., and Baron, L., "Kinematic inversion of functionally-redundant serial manipulators: application to arc-welding", *Transactions of the Canadian Society for Mechanical Engineering*, 2005, Canada.
- [12] Honegger, M. and Codourey, A., Redundancy resolution of a cartesian space operated heavy industrial manipulator, *IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 2094-2098, 1998.
- [13] Baron L., "A joint-limits avoidance strategy for arc-welding robots", *International Conference on Integrated Design and Manufacturing in Mechanical Engineering*, Montreal, Canada, May 2000.

CHAPTER 7

APPLICATIONS

"False facts are highly injurious to the progress of science, for they often endure long; but false views, if supported by some evidence, do little harm, for every one takes a salutary pleasure in proving their falseness." –

Charles Darwin

CHAPTER 7. APPLICATIONS

7.1. INTRODUCTION

Prior to the motivation of the present thesis, there was a permanent partnership agreement between the IDF and the official NX™ dealer in the Valencian Community (Avantek, currently converted to Procue). Therefore, leaving aside the standard applications on CN milling centres, NX™ provided an open environment for CAD/CAM/ROB integration research. All possible applications of robotics in machining industry started to be investigated by means of the KUKA™ complex robotic workcell introduced in Chapter 2.

After studying the RRS for such a workcell and the implementation of the treatment to give to the code generated by the CAM, the present Chapter deals with two practical cases studied at the IDF. As first case studied, the workcell is intended to machine a full 8x13 meters orographic model of a reservoir in the Mijares River (Spain), and afterwards the applicability of the CAM/ROB integration into traditional processes is tackled in the second case studied, in partnership agreement with the *Comité de Artistas Falleros de Valencia*.

At next section, a brief presentation of the specific materials used to carry out both works is done. A guideline of the specific treatment of the CAD file in each case studied is also highlighted.

7.2. MATERIAL AND METHODS

Besides the specific treatment of the CAD file imported to the NX™ platform, some common guidelines about the material (distinguishing between software and hardware) and methods used in both cases studied are exposed in this Section.

7.2.1. Material

i) Software

In addition to the NX™ platform, KUKA™ provides a basic graphical simulator which helps the operator when preparing a milling task. As first attempt, this software (Robomove™, Qdesign [5]) can be

profitable to visualize postures and possible interferences, validating the result of the RSS implemented (Figure 7.1).

Nevertheless, NXTM itself has a graphic simulator. NXTM-Motion is a CAE package integrated within NXTM allowing the kinematic simulation of complex mechanisms modelled or imported to the CAD module [2], like the exposed workcell. This package has been tested with good results and with the aimed advantage that the CAD and CAM work done is profitable within the same software. The input for NXTM-Motion is a two columns matrix containing a time counter and the associated joint values. However, it is still more cumbersome to manage due to the large weight of the CAD parts when compared with the lower weight of the parts managed by RobomoveTM. Different views of the workcell simulated in Motion and RobomoveTM are shown in Figure 7.1.

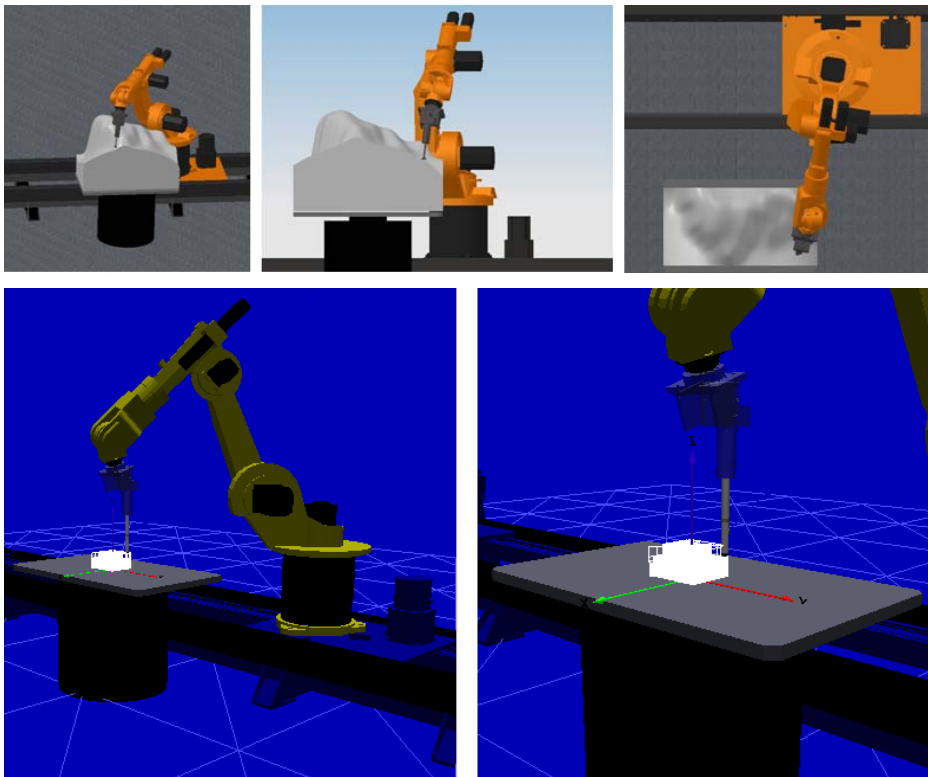


Figure 7.1. Up, different views of the workcell simulated in NXTM MOTION; down, two views of the simulation done with RobomoveTM.

ii) Hardware

Apart from the KUKA™ workcell (with the KRC2 controller [4]) used for the scope of this thesis and introduced in Chapter 2, some specific tools designed at the IDF were utilized to carry out the milling tasks, namely: the milling tool holder on the robot flange, and the workpiece holder on the rotary table.

The geometry of the milling tool (and the tool holder) was described in Chapter 2, and it has been made from a 15 mm aluminium sheet as shown in Figure 7.2-left. However, and due to the fact that CAM-software only consider the tool as revolute shapes (due to axial symmetry of the tool, see Chapter 4), it is modelled according to Figure 7.2-right. It gives the criterion of highest possible safety due to the fact that NX™ calculates the toolpath while detecting possible collisions of the tool holder modelled [1][2]. In fact, the orientation that the current tool could take regarding its +Z axis is not determined in NX™ at first, but fixed in the DH-model of the workcell (and consequently, to practical effects, in the measurement of the tool within the KRC2 controller).

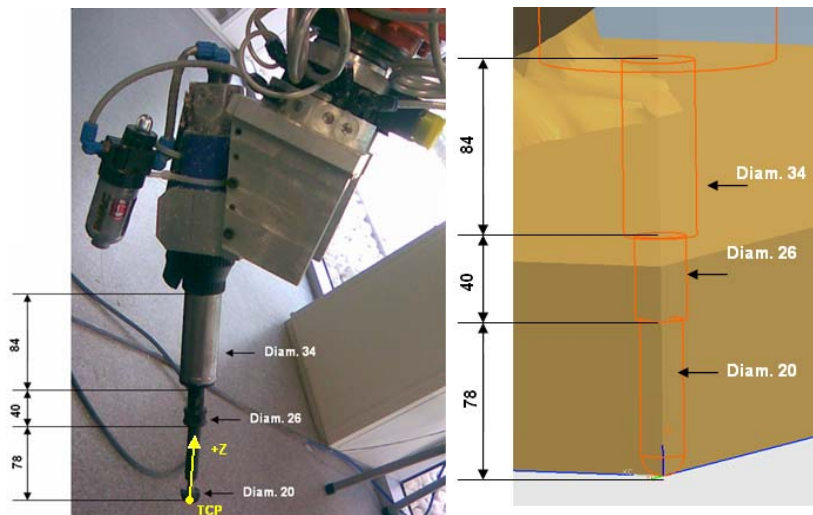


Figure 7.2. Left, real tool: an air turbine moves a 20 mm diameter spherical-tip tool; right, revolute model in NX™.

The workpiece holder (on the rotary table, Figure 7.3) fixes it during the milling work. Like the milling tool, it is also fed by means of a pneumatic system.

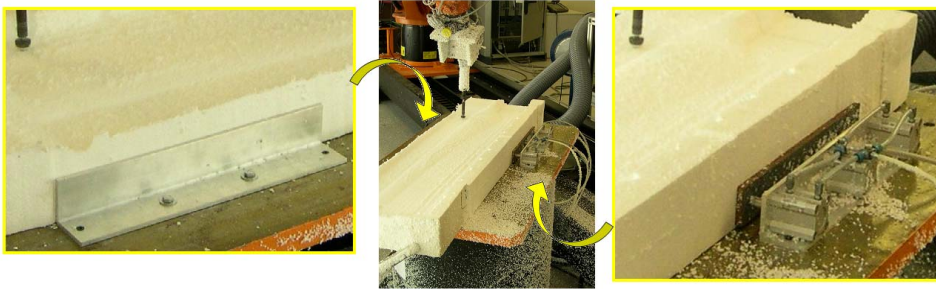


Figure 7.3. A pressurized air system (right) pushes the pistons against the opposite angle (left) to fix the workpiece.

When describing the materials utilized for the case studied, a special mention must be done about the blank composition. In this thesis, expanded polystyrene (EPS) was chosen. There are several advantages of modelling with EPS: it is a cheap material, it is also easier and quicker to machine, more stable over time, and less prone to damage than other materials. If needed, it is easy to divide the model into separate sections, and chop and change various elements. Finally, EPS can be treated with many different surface finishes and so can provide a more realistic model.



Figure 7.4. Left, EPS blanks; right, machining process of one piece in EPS.

7.2.2. Methods

The designs are developed by means of the NXTM-CAD module, in some cases after importing the original data source file. In these cases, any discontinuity or defect is restored. It is remarkable that the quality of the CAD model always determines the efficiency of the results that could be obtained later

in the following steps of the manufacturing process (Figure 7.5). Therefore, this is always the point of departure for applying the rest of computer assisted technologies.

With the cases studied, particular attention is to be paid for the very common processes starting with a CAD translation (namely, coming from different CAD systems) or a digitalization process. For each case studied, the particularities during the processing are explained at subsequent sections.

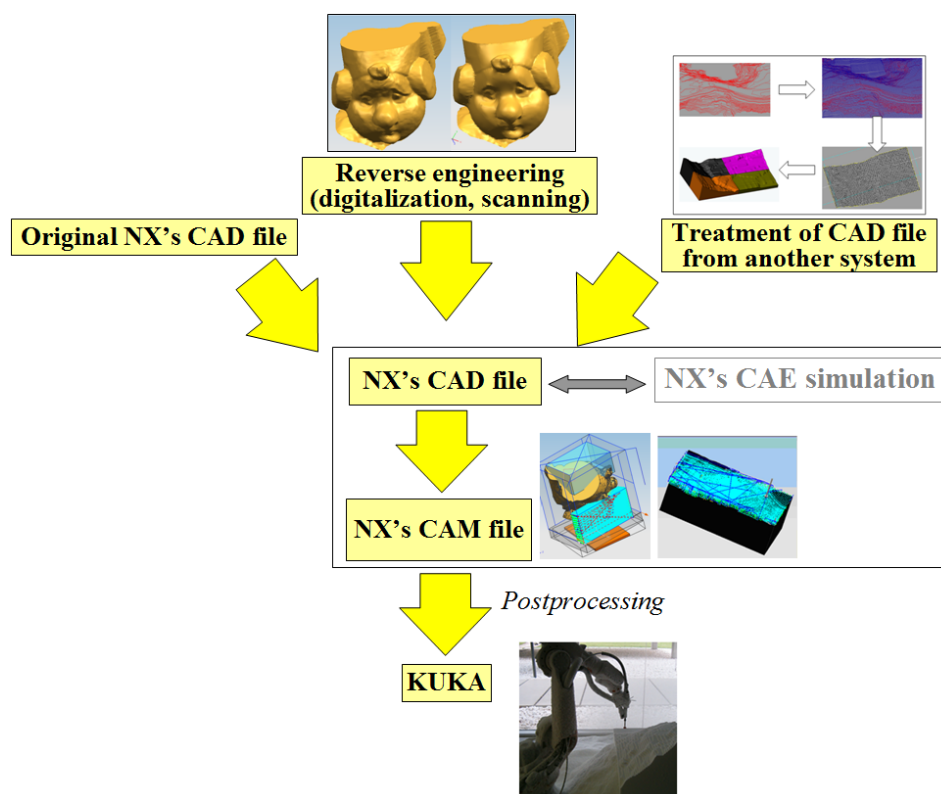


Figure 7.5. Flow process for the cases studied

Finally, as explained in Chapter 2, KUKA™ KRC2 control cabinet contains all the components and functions required to operate the robot. However, this cabinet is oriented to *industrial* environments so the control software is normally protected, having a limited access during processing a work. In fact, the externally programmed KRL codes (.src) must be compiled before its execution by updating them in a specific folder (*KRC:\RI\Program*). In addition,

this cabinet, more oriented to pick-and-place tasks, has a limited memory of 3 Mb.

To avoid both limitations, the more convenient solutions consists of executing a generic program in the specified folder (thus previously compiled) but reading the sequence of *CP* points from an external *.dat* file, as done with the *extended* version of the RobomoveTM package [5].

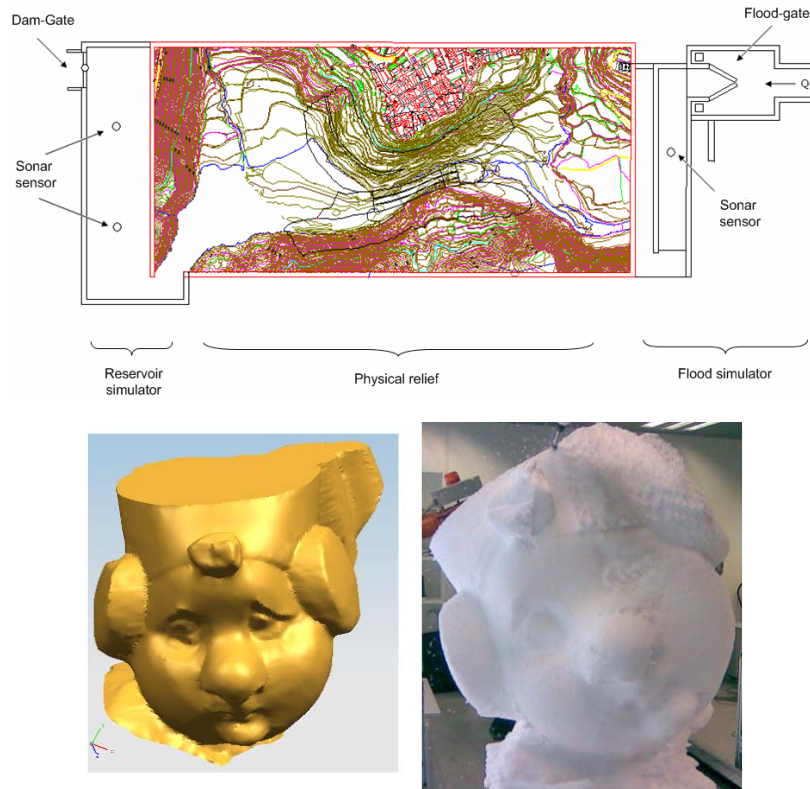


Figure 7.6. Up, sketch of the scaled model (factor 1:75) of the reservoir used to simulate flows, refluxes and water retentions; down, Valencian *Falla*.

7.3. CASES STUDIED

With the aim of validating the postprocessor designed, two cases are studied in the workcell of the IDF.

- In the first case, the workcell is intended to machine a full 8x13 meters *orographic model* of a reservoir in the Mijares River (Puebla de Arenoso, Spain), with the aim of simulating water avenues in order to study the

position of the wave that is formed when this avenue collides with the water stored in the reservoir (Figure 7.6, up).

- In the second case, the workcell is devoted to machine an EPS carving (namely, a Valencian *Falla*) being part of a partnership study with the *Comité de Artistas Falleros de Valencia* to evaluate the applicability of the current CAM/ROB integration into traditional processes (Figure 7.6, down).

7.3.1. Orographic model

Due to the dimensions of the model, it is obtained by assembling 120 blocks of 1x1x0.5 meters of EPS. The design is developed by means of the NXTM-CAD module after importing the original data source file from AUTOCADTM with *MDTTM v4* (the most commonly used system in topography). The contour lines must be fixed and then a surface mesh is interpolated. Then, this surface is divided to obtain the blocks (Figure 7.7).

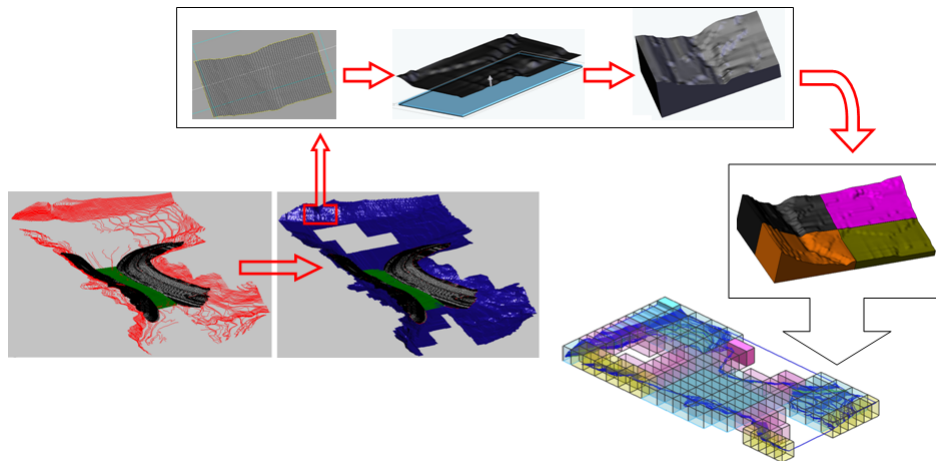


Figure 7.7. The model is obtained by assembling 120 blocks of 1x1x0.5 meters of EPS, after fixing the contour lines and interpolating mesh for each block.

Due to the relatively simple geometry of the blocks and the length of the tool (Figure 7.2), this first case studied was profited to validate the postprocessor in 3-axis milling operations (used for both cavity milling and surface finishing), previous to the second case studied (see Figure 7.8).

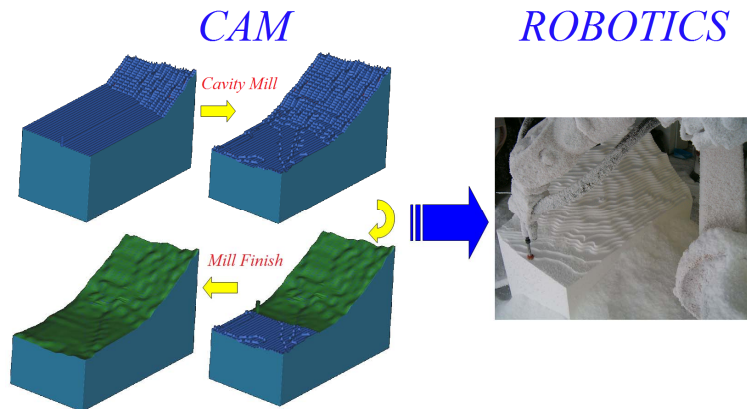


Figure 7.8. CAM/Rob process for the construction of each block, validating the postprocessor for 3-axis milling operations.

Thus, following the previous CAD process to obtain each block, the CAM process is summarized as follows:

i) Trajectory generation

After introducing the cutting parameters exposed in Chapter 4, both trajectories for Cavity Mill and Mill Finish operations are generated in NX, as it would be done for a conventional CN-machine, Figure 7.9.

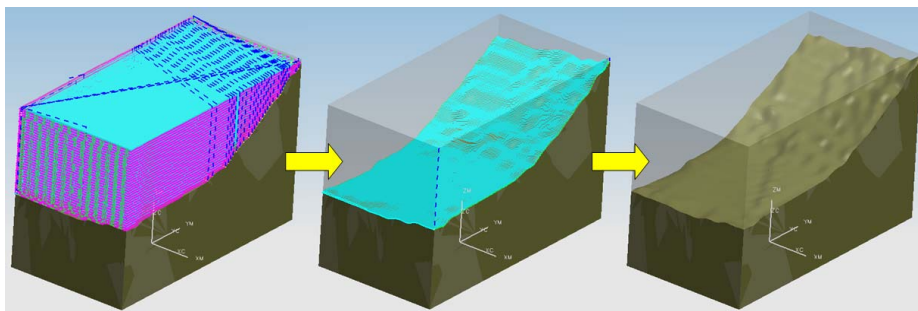


Figure 7.9. Trajectories for Cavity Milling and Finish Milling are generated in NX.

ii) Trajectory postprocessing

Figure 7.10, left, shows the generated toolpath directly passed to the graphical simulator RobomoveTM for a specific initial position. It can

be appreciated that the end of the workpiece could not be reached without moving the additional external joints (E1 and E2), and so the trajectory is red-coloured in that part (Figure 7.10, left). Thus, the same toolpath is postprocessed with the algorithms implemented and simulated within Robomove™ (Figure 7.10, right).

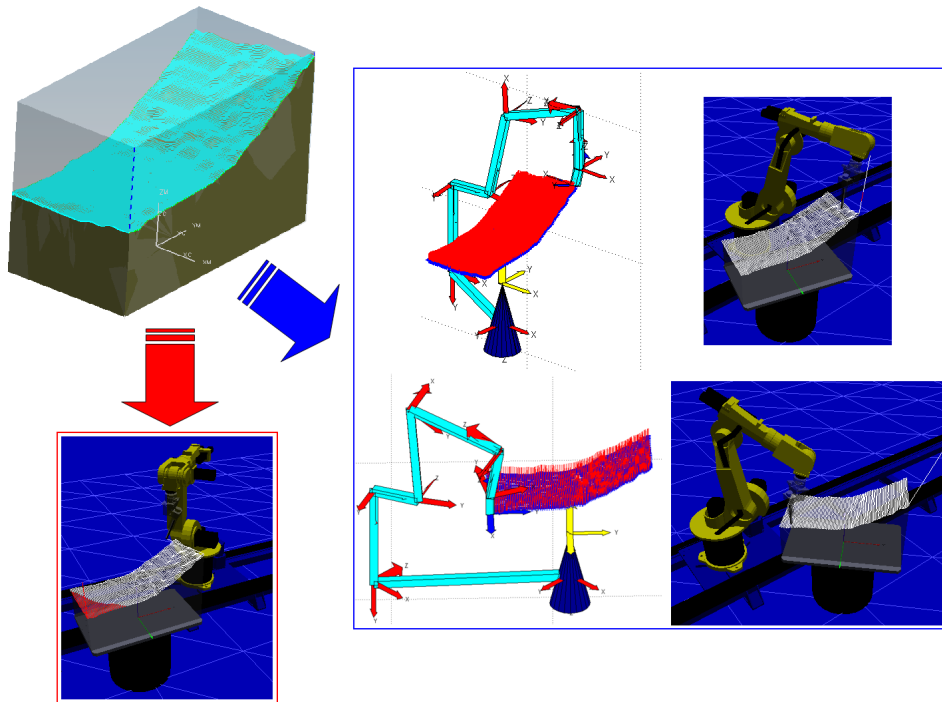


Figure 7.10. With the programmed algorithm, the additional joints are moved to reach the complete toolpath while maintaining a well conditioned posture.

As it can be appreciated, all joints are maintained between the allowable limits, while the condition number is kept between reasonable values.

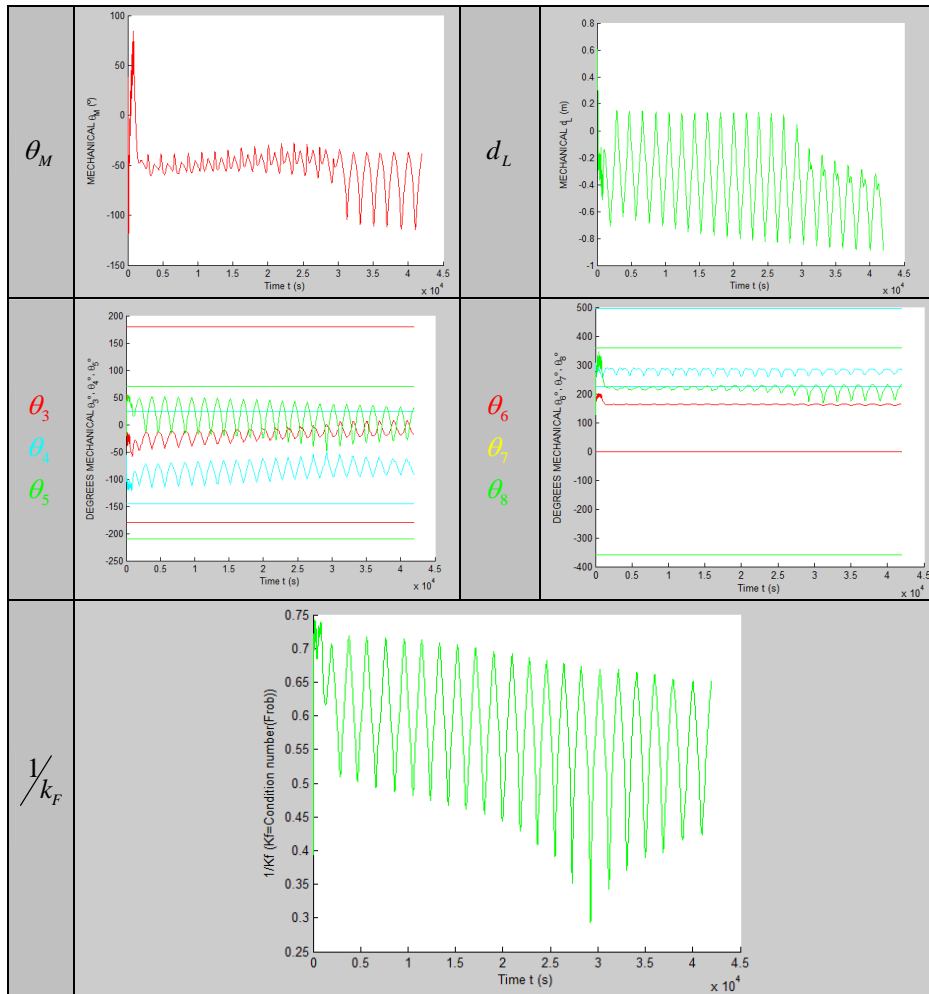


Table 7.1. Range of motion and conditioning of the manipulator while the execution of the task.

After milling all the blocks of the model, the final result was placed at the Hydraulic Engineering Department (DIHMA) of the Universidad Polit cnica de Valencia (Figure 7.11).



Figure 7.11. Final model in EPS with scaled factor 1:75 for flowing simulation (real dimensions of 8x13 m).

7.3.2. Valencian *Ninot*

The *Fallas* are a Valencian traditional celebration in praise of St. Joseph in Valencia, Spain (on March 19th). The term *Fallas* refers to both the celebration and the monuments created for the celebration. Prior to the celebration, much time has been spent preparing the *ninots* (namely, *puppets* or *dolls*) that are assembled to compose the *Falla* (Figure 7.12).

The *ninots* and their *falles* are developed according to an agreed upon theme to be a satirical joke at anything drawing the attention of the critical eyes of the celebrants (*fallers*). In modern times, this celebration has spawned a huge local industry (*Ciutat fallera*) where the artists elaborate the constructions with EPS, wood, paper and wax.



Figure 7.12. *Valencian Falla* composed of fanciful *ninots* in outrageous poses arranged in a gravity-defying architecture.

For the purpose of this case studied, the model of a *ninot* was provided by the *Comité de Artistas falleros de Valencia*. This model was given as a stereolithography (.stl) original file, a standard in scanning software. First of all, it had to be enlarged with a scale factor of 12,2:1 (based on particular requirements of the real *Falla*). This is due to the fact that most models are first done on a small size by traditional skills for its later digitalization. Then, it seemed profitable the smoothing of some regions to get a better milling result. This softening was carried out within the NX's CAD interface with the appropriate plug-ins for the treatment of faceted bodies (Figure 7.13).

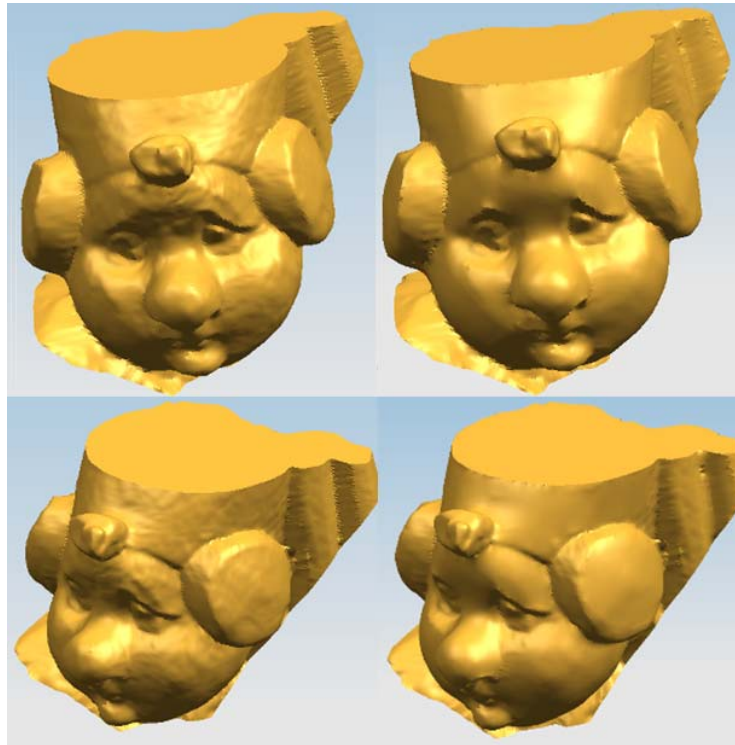


Figure 7.13. From left to right, the effects of the softening operation are shown. It has great relevance as it determines the rest of operations until the final milling.

The analysis of the geometry of the workpiece, its dimensions and the geometry of the tool determine number of milling operations to be planned and the related parameters.

Cavity milling operations are always done with a constant orientation to avoid the collision of the tool with the walls being generated between different levels (Figure 7.14). Thus, they consist of a series of 3-axes operations with the convenient tool orientation (namely, 3+2 milling operations, Figure 7.15 and Figure 7.16), but following the pattern of Section 7.3.1.

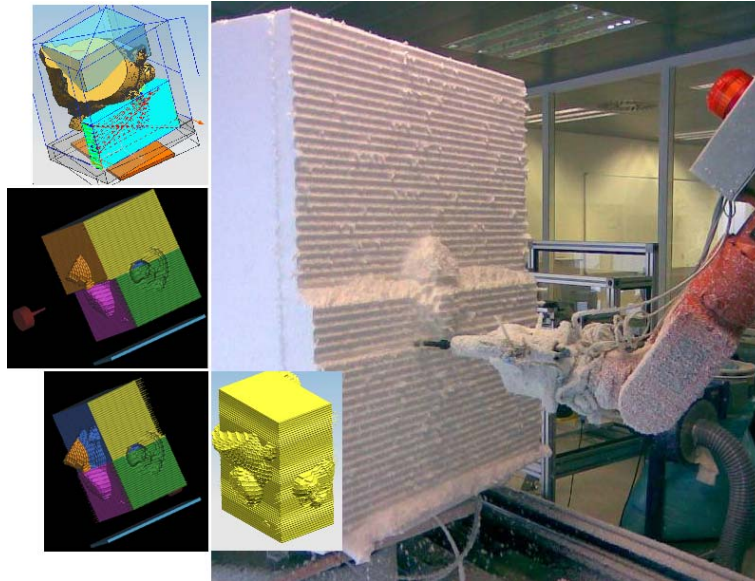


Figure 7.14. For the cavity milling, the workpiece is necessarily divided in different cutting areas (upper and lower zones). This treatment optimizes the use of the additional joints, and is strongly dependent on the tool's length.

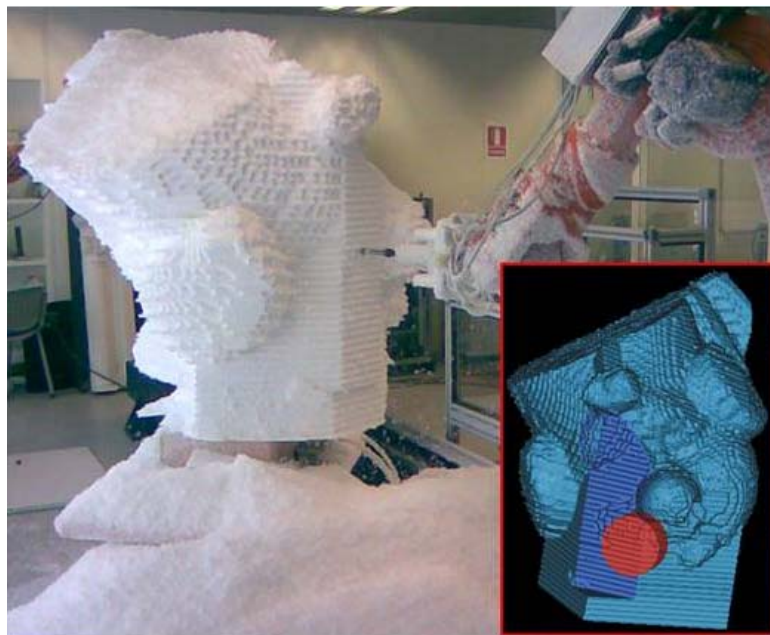


Figure 7.15. The successive cavity mill operations will be carried out with variable orientation of the tool, in order to reach all the parts of the *ninot*.



Figure 7.16. The attachment of the blank directly over the table makes the access to the lower parts difficult. Therefore, the blank is fixed by means of an intermediate piece which raises the height.

For comparison purposes, this section now focuses on a 5-axes milling operation on the ninot's surface. The toolpath is shown at Figure 7.17, in which the tool orientation is defined as normal to the surface along the tracking. Care must be taken in this case, where sudden changes must be prevented mainly due to surface defects from the scanning. Thus, not only the CAD treatment but also the graphical simulation is very supportive in these cases¹.

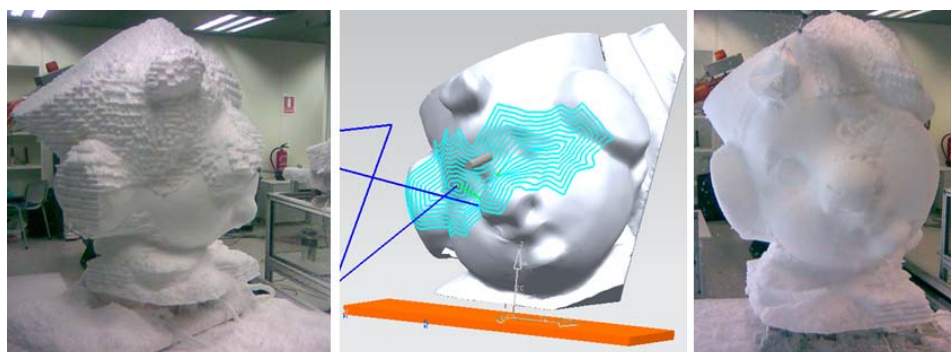


Figure 7.17. A 5-axes toolpath is planned on the eyes to test the postprocessor.

¹ It is noteworthy that, in many cases and to practical effects, it is preferable the use of a spherical-tip tool (namely, a *ball-nose end mill*) to get the same result than with variable orientation but with a 3+2 operation, leaving aside the profuse revision because of the visually negligible defects in the scanned surface.

For this experience, the same initial posture (HOME) is considered, namely

$$\{\theta_M, d_L, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\} \equiv \{0^\circ, 0.5 \text{ m}, -90^\circ, -90^\circ, 0^\circ, 0^\circ, 70^\circ, 0^\circ, 0^\circ\}$$

and the same task is attempted without and with the implemented postprocessor (Figure 7.18). Clearly, it can be appreciated much better performance within the second case.

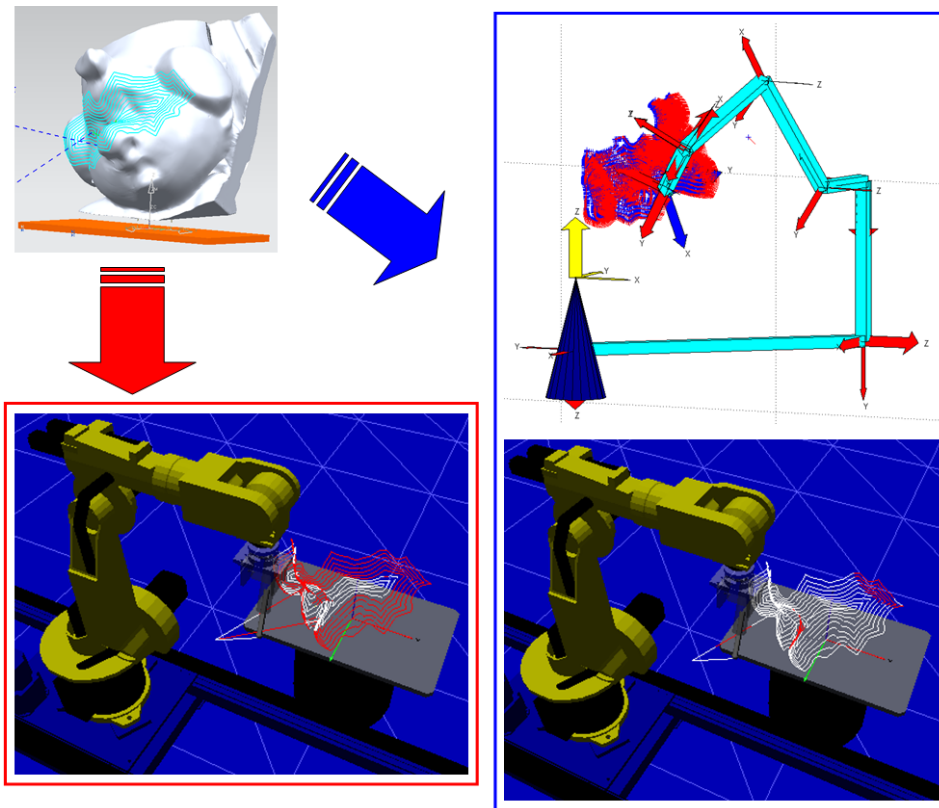


Figure 7.18. With the programmed algorithm, the additional joints are moved to obtain a better performance while maintaining a well conditioned posture.

As it can be appreciated, all joints are maintained between the allowable limits, while the condition number is kept between reasonable values.

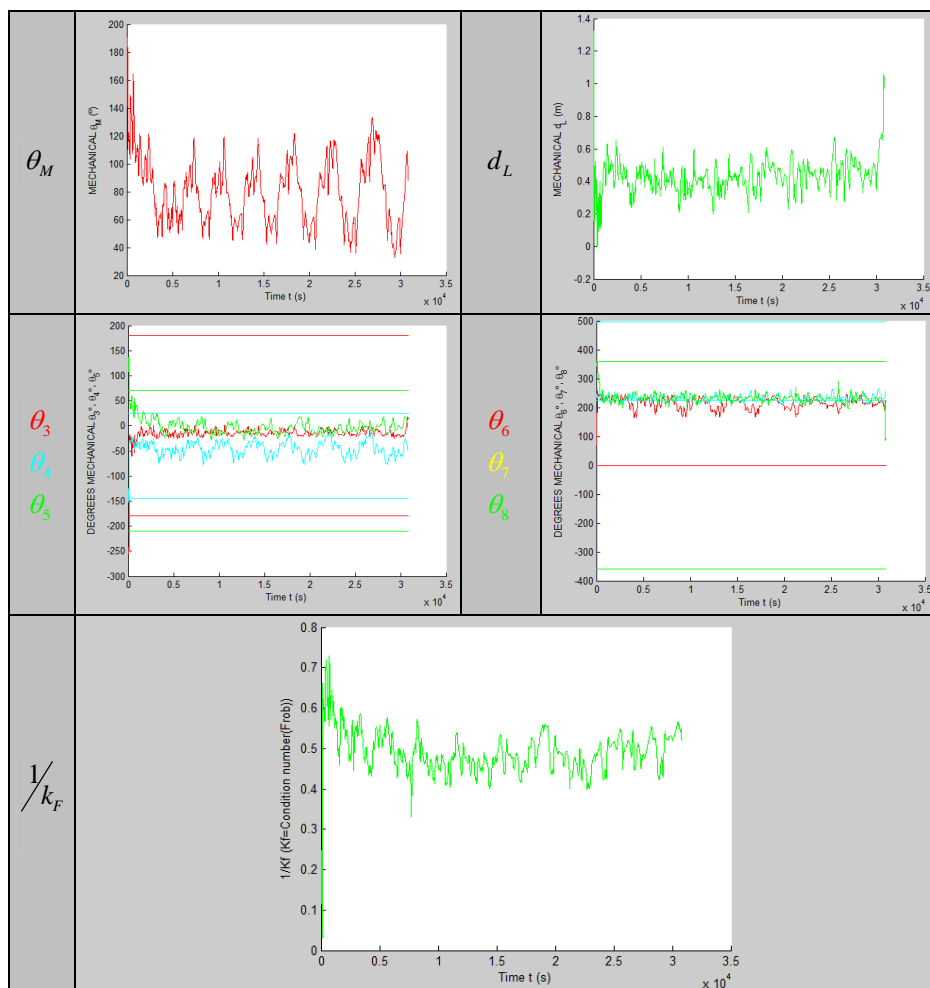


Table 7.2. Range of motion and conditioning of the manipulator while the execution of the task.

It can be considered a valuable result due to the fact that the limitations are found only in those surfaces where the orientation changes rapidly (Figure 7.19, up). As stated previously, special care must be taken in those cases.

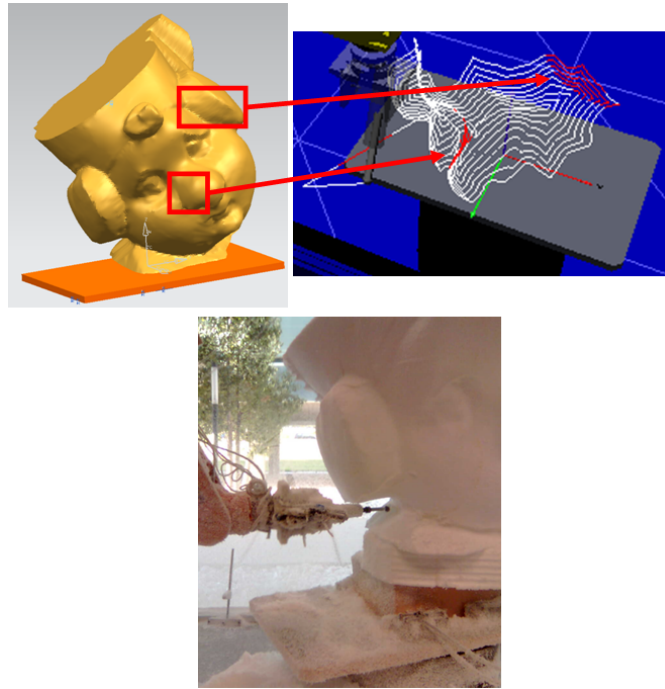


Figure 7.19. In some regions, where the orientation of the surfaces changes rapidly, the orientation of the tool associated to them can be problematic not only for the fast reaction of the posture required but also for the collision of the tool itself.

Although the case studied demonstrate the better performance achieved, to practical effects it is preferable the use of a spherical-tip tool (see note 1). The same consideration can be done when gaining access to regions where the surface is not a good reference for orientation (Figure 7.19, down).

Therefore, the experience could be repeated with a 3+2 operation. In this case, after estimating the most convenient orientation of the tool axis, it can be appreciated that the full surface could be machined almost completely (Figure 7.20).

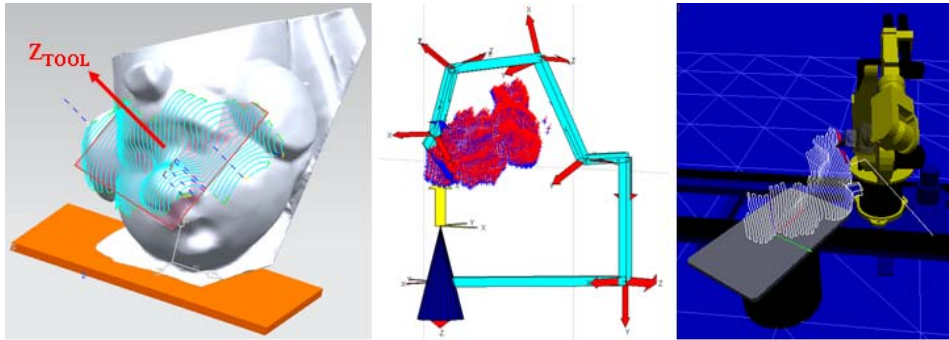


Figure 7.20. After estimating the most convenient Z_{TOOL} axis (left), the full surface could be machined almost completely (right) with a 3+2 milling operation.

REFERENCES

- [1] A. González, J. C. Sánchez; "CAM UG, Manual cam completo v2.1."; UGS Corporation 2005
- [2] "NX Documentation" ; ({\$UGII_base_dir}\UGDOC)
- [3] "Post Builder 5.0: Post Building Techniques"; UGS Corporation 2007
- [4] "KUKA System Software (KSS): Programación por el experto (KRC2 / KRC3)", Release 5.2., KUKA Corp., 2005.
- [5] "CAD-CAM off line programming for industrial robots", ROBOmove on line help v. 2.0, Qdesign S.r.l., 2007

CHAPTER 8
CONCLUSIONS AND FUTURE WORK

CHAPTER 8. CONCLUSIONS AND FUTURE WORK

8.1. CONCLUSIONS

After introducing the capabilities of an *industrial* serial robotic workcell, the present thesis has been focused on the postprocessing of the information generated by NXTM-CAM platform towards the KUKATM KRC2 controller for the workcell set at the IDF. Next, the most relevant conclusions and contributions of the present thesis are exposed:

- Both direct and inverse kinematic problems at the displacement and at joint-rate level have been obtained for a redundant workcell, contextualised in the postprocessing step from a CAM to the particular requirements of an industrial workcell (Chapter 2).
 - At the displacement level, due to redundancy, a known value for the external additional joints must be supposed in order to solve the inverse kinematic problem. With this consideration, a geometric resolution has been described as the best chance for a fast computation (Chapter 2).
 - Also the model at joint rate level has been documented, developing the concept of geometric Jacobian for the workcell studied. With it, the *condition number* of the *Homogeneous* Jacobian has been described as criteria to evaluate the efficiency of a manipulator posture. The characteristic length of the KUKATM KR15/2 has been obtained for this purpose.
- To study in depth in the architecture of the workcell, also the singularities that affect the workspace have been characterized with a geometric interpretation (Chapter 2).
- A due to the particular requirements of the *industrial* KUKATM workcell, which is assembled *in situ* and it is programmed by moving preferably the last six joints (see Chapter 4), a *Non-contact Planar Constraint Calibration Method* has been developed in Chapter 3 for the calibration of the external additional joints. It uses a three planar pattern fixed at the corresponding workplace, and a laser displacement sensor, and uses a *Non-Linear Least Squares* (NLSQ) procedure based on the sensor readings. It can be implemented

autonomously and is suitable for *on-site* calibration in an *industrial* environment at regular intervals, in contrast with other *open-loop* methods requiring extensive human intervention and expensive or demanding devices

- From the previous model at displacement level, a first approach to control the redundant joints has been done by means of a *fuzzy inference engine* integrated within the CAM system. It analyses the convenience of moving the additional joints like a skilled operator would do. Nevertheless, due to the high non-linearity and complexity of the model, and the variety of milling operations that exists, a growing number of rules for complex milling makes cumbersome the control (Chapters 4 and 5).
- The managing of the additional external joints and the redundancy due to the symmetry of the cutter tool: a functional postprocessor have been programmed inside the CAM system for the control of the redundancies at milling tasks. It is also expected to be easily applicable not only on any industrial robot, but also for different applications such as welding or painting labours.
- At joint rate level, where the problem becomes linear, the management of redundancies, with the use of a *Redundancy Resolution Scheme* (RSS), has been discussed. Previously, different types of redundancy (*functional* and *intrinsic*) have been identified in the workcell (Chapter 5)
- Different RRS were exposed in Chapter 5. Also the convenience of adjusting the weighting matrix which balances the joint behaviour in the achievement of secondary tasks is exposed. For that, a *fuzzy inference engine* is implemented to automatically tune this matrix, according to the actual robot posture. The secondary tasks mentioned above can be resumed as the maintenance of a well-conditioned posture and the avoidance of joint-limits by the maintenance of a reference posture.
- Previously selected RRS (*Virtual Joint Method* VJM and *Twist Decomposition Method* TDM) are implemented and tested in Chapter 6. VJM shows a more desirable behaviour (more stable and with a better conditioning during the trials done). This RRS deals with the *functional redundancy* (the one due to the symmetry axis of the cutter) by considering an additional (virtual) joint in the referred symmetry axis.
- VJM behaviour is improved by tuning the weights as introduced above, with a programmed fuzzy inference engine.

- Taking into account the current requirements of a challenging milling task (a continuous sphere), the VJM programmed greatly improves its performance by periodically using the inverse kinematic problem at displacement level. It periodically sets a point of departure with a significantly much better conditioned posture (Chapter 6).
- A functional postprocessor have been programmed inside the NXTM-CAM, improving the communication between software and the KUKATM robotic workcell.
- The implemented postprocessor is proven first in graphical simulation and then with real milling tasks at Chapters 6 and 7, by setting the additional external joints as required by the resolution of the implemented RRS. Two works, an *orographic model* (3-axes milling) and a *Valencian Falla* (5-axes milling) are described with special attention to the complete flow procedure of the CAD/CAM/ROB operations.

Therefore, the following reflections can be added:

- With the previous described work, the cycle in which the data generated by a CAM system are translated into a directly understandable language for an industrial controller is closed. In addition, the motion of the external additional joints (linear track and rotary table) and the spin on the tool symmetry axis is automatically reconsidered on the basis of a set of rules derived form skilled experience.
- It is also worth mentioning the great influence on the result of the performance vector h that tunes the secondary tasks (hence, also the relevance of the weighting matrix).
- The practical cases studied validate the effectiveness of these production systems for the milling of large prototypes with the use of soft materials.

8.2. FUTURE WORK

Next, some directions for future works are considered:

- Further study may be done with other RRS: the Schemes using the *Damped Least-Squares (DLS-) Inverse* have been rejected at first due to the fact that, clearly, the error introduced affects the precision at milling tasks. However, comparison with a Scheme directly using a *Weighted Pseudo-Inverse* in the same conditions is the next study to be published.

- The modus operandi applied explodes the capabilities of commercial CAM systems and the industrial robot controllers. It is expected to be easily applicable on any industrial robot configurations if the need arises, by the same guidelines. Thus, further investigation is to be done with different configurations and systems.
- With the same guidelines, this postprocessor is expected to be easily profitable not only on any industrial robot, but also for different applications tracking a toolpath, such as welding or painting labours.
- As stated in Chapter 2, the best conditioned posture, the best the robot behaves the with regard to *force* (and motion) transmission. A Force/Torque sensor has been acquired by the IDF to be mounted in the robot flange. Further implementation must be done to tune the speed of the TCP during the tracking accordingly to the sensed efforts, mostly at weak conditioned postures. First attempts with OPC (*Ole Process Control*) technologies were not profitable, but new modules have been acquired to KUKA™.
- Without losing sight of the importance of obtaining a more efficient expert control, it is also interesting to have a tool to visually validate the motions of the robotic manipulator. As mentioned before, NX™ is a powerful CAD system in which the workcell has been modelled. Further efforts may be done to lighten the CAD-parts as done in Robomove™.
- Commercial robot controllers, more prepared for *pick and place* or *assembly* tasks, suffer from lack of memory when a long milling program is to be executed. The implementation of a buffer may override this problem, but at the same time this problem is solved in the incoming new robots.

APPENDICES

A.1. IK POSITIONING PROBLEM. MATLAB CODE.....	271
A.2. CALIBRATION OF THE ADDITIONAL EXTERNAL AXES	274
A.3. TCL CONCEPTS FOR THE PROGRAMMING OF A POSTPROCESSOR IN THE PLATFORM NX	285
A.4. CHARACTERISTIC LENGTH L OF THE KUKA KR-15/2 (CORRESPONDING TO SECTION 2.4.6.).....	304
A.5. MATLAB CODE FOR THE POSTPROCESSING OF CLSF FOUNDED ON THE VJM WITH PERIODIC RE-EVALUATION.	305

A.1. IK POSITIONING PROBLEM. MATLAB CODE.

The following lines resume the Matlab code solving the IKP of the IDF's KUKA workcell. Some comments are done directly on the code, but more precise understanding is achieved by following the explanations given in Section 2.4.3.

```
function q=IK_KUKA_periorizacion100s(thetamesa,dtrack,thetaVJM,T_CAMM)

% MOVIMIENTO GRUESO (t1 t2 t3)
tm=thetamesa-pi; % E2 (rad)
dl=dtrack; % E1 (m)
A_Base_a_TCP=T_CAMM*rotz(-thetaVJM); % SC TCP expresado en la mesa SC
Base;
A_Brida_a_TCP=rotx(0.3564)*transl(0,0,-0.1197); % SC TCP expresado en el
SC Brida
A_TCP_a_Brida=inv(A_Brida_a_TCP); % SC Brida expresado en el SC TCP

A_Base_a_Brida=A_Base_a_TCP*A_TCP_a_Brida; % SC Brida respecto de SC BAsE
B en la mesa

% Con la matriz recien calculada ya puedo hallar la posicion de la munyeca
% W respecto de la Base, por Pieper (hago el calculo en m)
Pos_Base_a_Wrist=([A_Base_a_Brida(1,4) A_Base_a_Brida(2,4)
A_Base_a_Brida(3,4)]'+0.4434*[A_Base_a_Brida(1,3) A_Base_a_Brida(2,3)
A_Base_a_Brida(3,3)]');

% Con esto, tengo las coordenadas de la munyeca en la MESA, pero quiero
% aplicar el metodo geometrico desde la base del robot $robroot. Primero
% hacer la rotacion y luego la trastacion de las coordenadas:
% a) ROTACION (en Z)
Pos_Base_a_Wrist_rotada=rotz(-tm)*[Pos_Base_a_Wrist(1) Pos_Base_a_Wrist(2)
Pos_Base_a_Wrist(3) 1]';
% b)TRASLACION
Pos_Robot_a_Wrist=[1 0 0 0.803;0 1 0 -dl;0 0 1 0.305;0 0 0
1]*Pos_Base_a_Wrist_rotada;

Mx=Pos_Robot_a_Wrist(1);
My=Pos_Robot_a_Wrist(2);
Mz=Pos_Robot_a_Wrist(3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MOVIMIENTO GRUESO (t1 t2 t3) %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% t1
t1=-atan2(My,Mx)+pi;

aprox_cero=1e-5;
% if abs(t1)<=aprox_cero
% t1=0;
% end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% t2 y t3
p=sqrt(Mx^2+My^2)-0.300;
h=(Mz-0.675);
```

```

eps=atan2(h,p);
a=sqrt(0.155^2+0.600^2);
b=0.650;
c=sqrt(h^2+p^2);
fi=atan(0.155/0.600);

if c>=(a+b)
    t3=0; % este es el límite de brazo estirado en el mecanizado (ver
dibujo articulo ICM)
    t2=eps; % el triangulo abc no existe
    warning('punto fuera del alcance, brazo estirado')
    q=q0;
    return
else
    s=(a+b+c)/2;
    r=sqrt((s-a)*(s-b)*(s-c)/s);
    alfa=2*atan(r/(s-a));
    sig=2*atan(r/(s-c));

    t2=-eps-alfa;
    t3=pi-sig+fi-pi/2; % resto pi/2 para trabajar modelo DH, no el angulo
mecanico KUKA
end

if abs(t2)<=aprox_cero
    t2=0;
end
if abs(t3)<=aprox_cero
    t3=0;
end

% MOVIMIENTO FINO (t4 t5 t6)

rol=-t1-pi; % resto pi por motivos del DH
Rzt1=rotz(rol);

ro2=pi/2+(t2+(t3+pi/2));
R_yprima_ro2=roty(ro2);

% con la composicion de estas rotaciones y la posicion de la muñeca
% referida a Robroot tengo la matriz homogenea T_RM que me define la
posicion y
% orientacion en ese punto M, osea T_RM= [[R_RM][Pos_Robot_a_Wrist]';0 0 0
1]
T_RM=Rzt1*R_yprima_ro2;
T_RM(:,4)=Pos_Robot_a_Wrist;

% Ahora voy a posicionar la brida
% (que la conozco respecto de la mesa como A_Base_a_Brida) respecto del
Robroot. Para
% ello hago lo mismo que hice con la muñeca (deshacer la rotacion de la
% mesa y la traslacion del lineal)
T_R6=[1 0 0 0.803;0 1 0 -d1;0 0 1 0.305;0 0 0 1]*rotz(-tm)*A_Base_a_Brida;
T_M6=inv(T_RM)*T_R6;

% Finalmente
t4=-atan2(-T_M6(2,3),-T_M6(1,3));
t5=atan2(sqrt(T_M6(3,1)^2+T_M6(3,2)^2),T_M6(3,3));

```



```
t6=-atan2(-T_M6(3,2),-T_M6(3,1));  
  
if abs(t4)<=aprox_cero  
    t4=0;  
end  
if abs(t5)<=aprox_cero  
    t5=0;  
end  
if abs(t6)<=aprox_cero  
    t6=0;  
end  
  
% theta  
q=[tm+pi d1 t1 t2 t3 t4 t5 t6];  
end
```

A.2. CALIBRATION OF THE ADDITIONAL EXTERNAL AXES

In this Matlab code, a *NLSQ* iterative procedure is carried out to obtain the errors $\Delta\beta_E$ in the assembly DH parameters of the additional external joints (linear track and rotary table) that are previously introduced on purpose, namely,

$$\Delta\beta_E = \begin{bmatrix} \Delta\alpha_{1E} \\ \Delta a_{1E} \\ \Delta\theta_{ME} \\ \Delta d_{1E} \\ \Delta\theta_{2E} \\ \Delta d_{LE} \end{bmatrix} = \begin{bmatrix} 0.01 \\ 0.06 \\ 0.07 \\ 0.05 \\ 0.01 \\ 0.08 \end{bmatrix} \quad (\text{mm, rad}) \quad (0.1)$$

It corresponds to the study done at Section 3.3., as follows:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% EXT JOINTS CALIBRATION %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
% format long g
format short g

deg2rad=(2*pi)/360; % multiplicador para pasar a rads
rad2deg=360/(2*pi); % multiplicador para pasar a degs
rotacion_de_brida_a_tool=rotx(pi);
traslacion_de_brida_a_tool=transl(0,0,250);
T_6LASER=traslacion_de_brida_a_tool*rotacion_de_brida_a_tool;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% KUKA MODEL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

syms tm E2 real; % para TOM (mm, t rads, E y A en °)
syms dl E1 real; % para TML
syms t1 A1 real; % para TL1
syms t2 A2 real; % para T12
syms t3 A3 real; % para T23
syms t4 A4 real; % para T34
syms t5 A5 real; % para T45
syms t6 A6 real; % para T56

DH_KUKA=[pi 0 0 305 0;
         pi/2 -803 0 0 1;
         -pi/2 0 0 -675 0;
         pi/2 300 0 0 0;
         0 650 0 0 0;
         pi/2 155 0 -600 0;
         -pi/2 0 0 0 0;
         -pi/2 0 0 140 0];

Q=[tm dl t1 t2 t3 t4 t5 t6];

```

```

T06=fkine(DH_KUKA,Q); % transformacion a la brida,

% offsets con las lecturas reales de la consola (modelo mecanico):
tm=deg2rad*E2;
dl=E1+2977.17;
t1=deg2rad*A1;
t2=deg2rad*A2;
t3=(deg2rad*A3)-pi/2;
t4=deg2rad*A4;
t5=deg2rad*A5;
t6=(deg2rad*A6);

T06_KUKA=subs(T06);
Q_KUKA=[E2 E1 A1 A2 A3 A4 A5 A6]; % parametros de articulacion en degs y
mm

% Solo consideramos la posicion (x,y,z)' (no medimos orientacion)
Pideal=[T06_KUKA(1,4),T06_KUKA(2,4),T06_KUKA(3,4)]';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% suponer errores en todos los parametros de montaje de los ejes
% externos (altura d1, alejamiento a1, inclinacion lateral del rail
alpha,
% inclinacion frontal del rail theta2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
syms d1 delta_d1 a1 delta_a1 alpha delta_alpha theta2 delta_theta2
delta_E2 delta_E1 real
syms tm E2 real; % para TOM
syms dl E1 real; % para TML
syms t1 A1 real; % para TL1
syms t2 A2 real; % para TL2
syms t3 A3 real; % para T23
syms t4 A4 real; % para T34
syms t5 A5 real; % para T45
syms t6 A6 real; % para T56

DH_KUKA_con_delta_beta=[pi 0 0 (305+delta_d1) 0;
(pi/2+delta_alpha) (-803+delta_a1) (0+delta_theta2) 0 1;
-pi/2 0 0 -675 0;
pi/2 300 0 0 0;
0 650 0 0 0;
pi/2 155 0 -600 0;
-pi/2 0 0 0 0;
-pi/2 0 0 140 0];

T06_con_delta_beta=fkine(DH_KUKA_con_delta_beta,Q); % transformacion al
efector final

% offsets con lecturas "reales" de la consola KUKA, y los deltas de los
ejes externos para hallar
% el posible error:
tm=deg2rad*(E2+delta_E2);
dl=E1+2977.17+delta_E1;
t1=deg2rad*A1;
t2=deg2rad*A2;

```

```

t3=(deg2rad*A3)-pi/2;
t4=deg2rad*A4;
t5=deg2rad*A5;
t6=(deg2rad*A6);

T06_KUKA_con_delta_beta=subs(T06_con_delta_beta); % transformacion al
efector final
Q_KUKA=[E2 E1 A1 A2 A3 A4 A5 A6];

% Para este ajuste, considero la posicion (x,y,z)' sobre la mesa y la
% orientacion ABC sobre la mesa
P_con_delta_beta=[T06_KUKA_con_delta_beta(1,4),T06_KUKA_con_delta_beta(2,4
),T06_KUKA_con_delta_beta(3,4)]';
r_con_delta_beta=T06_KUKA_con_delta_beta(1:3,1:3);

PO_LASER_con_delta_beta=eval(T06_KUKA_con_delta_beta)*T_6LASER;

% Modelo con supuesto de error (para comprobar la convergencia, supongo
estos errores a los
% que deberá converger):
% - error de altura d1: 0.05 mm
delta_d1=0.05 % 0;
% - error de inclinacion lateral del rail alpha1: 0.01 rad (= 0.57°)
delta_alpha1=0.01 % 0;
% - error de alejamiento a1: 0.06 mm
delta_a1=0.06 % 0;
% - error de inclinacion frontal del rail theta2: 0.01 rad (= 0.57°)
delta_theta2=0.01 % 0;
% - error angular en el giro de la mesa: 0.07° (= 0.012 rad), en la
ecuacion
% lo entro en grados, siendo ahí pasado a radianes dentro del modelado de
la celula.
delta_E2=0.07 % 0;
% - error de desplazamiento del lineal: 0.08 mm
delta_E1=0.08 % 0;

T06_KUKA_con_error=subs(T06_KUKA_con_delta_beta);
P_con_error=subs(P_con_delta_beta);
r_con_error=subs(r_con_delta_beta);

PO_LASER_con_error=eval(T06_KUKA_con_error)*T_6LASER;

syms delta_d1 delta_a1 delta_alpha1 delta_theta2 delta_E1 delta_E2 real

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MEDICION DE PUNTOS DE CALIBRACION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% A) Puntos ideales (cinematica inversa ideal)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Vamos a trabajar con 3 planos ortogonales sobre los que barreré el laser
% con el programa de la carita. El objetivo es
% repasar las siguientes matrices de puntos en cada plano (cada una de 16
% puntos)

dimensiones_matriz_puntos_Pm_gen=[4,4];

```

```

numero_de_puntos_Pm_gen=dimensiones_matriz_puntos_Pm_gen(1)*dimensiones_ma
triz_puntos_Pm_gen(2);
espaciado_puntos_Pm=100; %mm

[M,N]=meshgrid(-
espaciado_puntos_Pm*((dimensiones_matriz_puntos_Pm_gen(1)-
1)/2):espaciado_puntos_Pm:espaciado_puntos_Pm*((dimensiones_matriz_puntos_
Pm_gen(1)-1)/2),-
espaciado_puntos_Pm*((dimensiones_matriz_puntos_Pm_gen(2)-
1)/2):espaciado_puntos_Pm:espaciado_puntos_Pm*((dimensiones_matriz_puntos_
Pm_gen(2)-1)/2));
cont_Pm=1;
for j=1:dimensiones_matriz_puntos_Pm_gen(1)
    for i=1:dimensiones_matriz_puntos_Pm_gen(2)
        Pm_gen(cont_Pm,:)=M(i,j) N(i,j)]; % Pm_generador
        cont_Pm=cont_Pm+1;
    end
end

% 16 Puntos en Plano XY (superficie mesa, un cuadrado de 200 + 200 mm)
ABC_xy=[0 0 0]; % este dato es para la IK que voy a hacer, y es la
posicion del tool con Z entrante
Pm_xy=[Pm_gen zeros(size(Pm_gen,1),1)]; %la tercera columna es 0,
porque el punto esta en el plano XY
% plot3(Pm(:,1),Pm(:,2),Pm(:,3)); hold;
for i=1:size(Pm_xy,1)
    Pm_xy_ABC(i,:)=Pm_xy(i,:) ABC_xy];
end

% 16 Puntos en Plano XZ (plano vertical paralelo al lado corto de la
mesa)
ABC_xz=[0 0 -pi/2]; % idem
Pm_xz=[Pm_gen(:,1) zeros(size(Pm_gen,1),1) Pm_gen(:,2)]; %la segunda
columna es 0, porque el punto esta en el plano XZ
for i=1:size(Pm_xz,1)
    Pm_xz_ABC(i,:)=Pm_xz(i,:) ABC_xz];
end

% 16 Puntos en Plano YZ (plano vertical paralelo al lado largo de la
mesa)
ABC_yz=[0 -pi/2 0]; % idem
Pm_yz=[zeros(size(Pm_gen,1),1) Pm_gen]; %la primera columna es 0,
porque el punto esta en el plano YZ
for i=1:size(Pm_yz,1)
    Pm_yz_ABC(i,:)=Pm_yz(i,:) ABC_yz];
end

Pm=[[Pm_yz_ABC];[Pm_xz_ABC];[Pm_xy_ABC]];

% Vamos a considerar dos casos de los ejes externos [E1 E2],
casos=[[-2000 0]; [-2400 45]]; % empiricamente hemos comprobado que esta
dentro del campo de trabajo

for cont_casos=1:size(casos,1)
    for i=1:size(Pm,1)
q_caso(i,:)=IK_KUKA_laser_cualquier_ori(Pm(i,1),Pm(i,2),Pm(i,3),Pm(i,4),Pm

```

```

(i,5),Pm(i,6),casos(cont_casos,1),casos(cont_casos,2))*rad2deg; %
coordenadas articulación A1-A6 para cada punto
end
q(cont_casos, :, :) = q_caso;
end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A) Puntos alcanzados
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Al robot le pedimos esos puntos y lo va a hacer mal por los
% errores anteriormente supuestos en los parametros de montaje de los ejes
% externos

% En primer lugar, con los siguientes bucles anidados voy a ver donde se
% situaría el
% hipotetico TCP del laser (recordemos que esta a 250mm de la brida)

for cont_casos=1:size(casos,1)
  for i=1:size(Pm,1)
    E2=casos(cont_casos,2);
    E1=casos(cont_casos,1);
    A1=q(cont_casos,i,1);
    A2=q(cont_casos,i,2);
    A3=q(cont_casos,i,3);
    A4=q(cont_casos,i,4);
    A5=q(cont_casos,i,5);
    A6=q(cont_casos,i,6);

    plotbot(DH_KUKA,[deg2rad*E2 E1+2977.17 deg2rad*A1 deg2rad*A2
deg2rad*(A3-90) deg2rad*A4 deg2rad*A5 deg2rad*A6],'flw');
frame(eye(4),'y',500);

    PO_LASER=eval(T06_KUKA_con_error)*T_6LASER;

    % ABC_alcanzada=KUKAtr2rpy(PO_LASER);
    Pos_alcanzada=subs(PO_LASER(1:3,4))';
    Vector_del_laser_respecto_BASE=subs(PO_LASER(1:3,3))';

    % Ps(cont_casos,i,:)=[Pos_alcanzada ABC_alcanzada]; % Para q(numero de
caso,punto Pm del plano de 1 a 75),articulacion) <--> Ps(numero de
caso,punto Pm del plano de 1 a 12),coordenada xyzABC de 1 a 6)
    Ps(cont_casos,i,:)=[Pos_alcanzada Vector_del_laser_respecto_BASE]; %
Para q(numero de caso,punto Pm del plano de 1 a 75),articulacion) <-->
Ps(numero de caso,punto Pm del plano de 1 a 12),coordenada xyzABC de 1 a
6)
  end
end

% Para hacer mis minimos cuadrados, la unica medición de que dispongo es
la
% la distancia que mide el laser (cuasi perpendicular al plano y cuanto
mas se corrijan lo errores del robot,
% mas cierta será esta perpendicularidad). VAMOS A MEDIR LA DISTANCIA QUE
LEE EL LASER que, en
% definitiva, quiero que sea 0.

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PROCEDIMIENTO DE CALIBRACION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% delta_P=J*delta_beta --> delta_beta=(J+).delta_P

syms d1 delta_d1 a1 delta_a1 alpha1 delta_alpha1 theta2 delta_theta2
delta_E2 delta_E1 real
syms tm E2 real; % para T0M
syms d1 E1 real; % para TML
syms t1 A1 real; % para TL1
syms t2 A2 real; % para T12
syms t3 A3 real; % para T23
syms t4 A4 real; % para T34
syms t5 A5 real; % para T45
syms t6 A6 real; % para T56

Q=[tm d1 t1 t2 t3 t4 t5 t6];
Q_KUKA=[E2 E1 A1 A2 A3 A4 A5 A6];

DH_KUKA_param=[pi 0 0 (d1) 0;
               (alpha1) (a1) (theta2) 0 1;
               -pi/2 0 0 -675 0;
               pi/2 300 0 0 0;
               0 650 0 0 0;
               pi/2 155 0 -600 0;
               -pi/2 0 0 0 0;
               -pi/2 0 0 140 0];

T06_param=fkine(DH_KUKA_param,Q);
PO_LASER_param=eval(T06_param)*T_6LASER;

tm=deg2rad*E2;
d1=E1+2977.17;
t1=deg2rad*A1;
t2=deg2rad*A2;
t3=(deg2rad*A3)-pi/2;
t4=deg2rad*A4;
t5=deg2rad*A5;
t6=(deg2rad*A6)+pi;

PO_LASER_KUKA_param=subs(PO_LASER_param);

D_LASER_KUKA_param=PO_LASER_KUKA_param(1:3,4); % fijate que, según el
plano, la D hace referencia a la coordenada X Y o Z de la base.

% Directamente obtengo Jr, dado que J completa es enorme

dhparam=[d1 alpha1 a1 theta2 E2 E1];
for i=1:size(dhparam,2)
    Jr(:,i)=diff(D_LASER_KUKA_param,dhparam(i)); % dependencia de la
posicion xyz respecto a cada parametro de DH a ajustar;
end

% ya estan sustituidos en Jr los parametros de dh que NO son variables y
que asumimos
% que no tienen error, osea que quedará en funcion de E2 & E1
% (por ser joints del robot) y tambien de d1, alpha1, a1, theta2 (que
consideramos que

```

```

% tienen error).

    d1=305+delta_d1;
    alpha1=pi/2+delta_alpha1;
    a1=-803+delta_a1;
    theta2=0+delta_theta2;
    E2=E2+delta_E2;
    E1=E1+delta_E1;

Jr_con_delta=subs(Jr);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ITERACION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
numero_de_iteraciones=20;

clear delta_d1 delta_alpha1 delta_a1 delta_theta2 delta_E1 delta_E2
V_correc_acumuladas V_correc_tras_cada_iteracion V_errores_posic

% inicialmente se supone no-error en los parametros de DH
delta_d1=0; delta_alpha1=0; delta_a1=0; delta_theta2=0; delta_E2=0;
delta_E1=0;
j=1;
for cont_casos=1:size(casos,1)
    for i=1:numero_de_puntos_Pm_gen
        syms DistXbase DistYbase DistZbase real

delta_P_filas(j,1)=eval(solve(Ps(cont_casos,i,1)+DistXbase*Ps(cont_casos,i
,4),DistXbase));
        delta_P_filas(j,2)=-
eval(solve(Ps(cont_casos,numero_de_puntos_Pm_gen+i,2)+DistYbase*Ps(cont_ca
sos,numero_de_puntos_Pm_gen+i,5),DistYbase));
        delta_P_filas(j,3)=-
eval(solve(Ps(cont_casos,2*numero_de_puntos_Pm_gen+i,3)+DistZbase*Ps(cont_
casos,2*numero_de_puntos_Pm_gen+i,6),DistZbase));

        j=j+1;
    end
end

for iter=1:numero_de_iteraciones

% OJO, reset en delta_P y delta_beta porque se calculan de nuevo, pero
% no delta_theta ni delta_a1 porque acumulan la correccion
clear delta_P delta_beta Dpi b

% los nuevos errores de posicion se calcularán de nuevo al iniciar el
bucle

delta_P_cols=delta_P_filas'; % en columnas
num_filas_delta_P_cols=size(delta_P_cols,1); % 3, D en Xbase, en Ybase y
en Zbase
num_columnas_delta_P_cols=size(delta_P_cols,2); % 50, la mitad son del
caso 1 y otra mitad del caso 2

for h=1:num_columnas_delta_P_cols
    clear aux

```



```

    aux=delta_P_cols(:,h);
    for k=0:(num_filas_delta_P_cols-1)
        delta_P(num_filas_delta_P_cols*h-k,:)=aux(num_filas_delta_P_cols-k,:);
    end
end

V_errores_posic(:,iter)=delta_P;

% damos forma a la matriz de observacion (Jacobiano, para cada punto
% ensayado), W, con Jr_con_delta

num_filas_Jr_con_delta=size(Jr_con_delta,1);

for cont_casos=1:size(casos,1)
    for i=1:numero_de_puntos_Pm_gen
        syms E2 E1 A1 A2 A3 A4 A5 A6 real
        clear J_o

        E2=casos(cont_casos,2); E1=casos(cont_casos,1); A1=q(cont_casos,i,1);
        A2=q(cont_casos,i,2);
        A3=q(cont_casos,i,3); A4=q(cont_casos,i,4); A5=q(cont_casos,i,5);
        A6=q(cont_casos,i,6);
        J_o(1,:)=subs(Jr_con_delta(1,:));

        E2=casos(cont_casos,2); E1=casos(cont_casos,1);
        A1=q(cont_casos,numero_de_puntos_Pm_gen+i,1);
        A2=q(cont_casos,numero_de_puntos_Pm_gen+i,2);
        A3=q(cont_casos,numero_de_puntos_Pm_gen+i,3);
        A4=q(cont_casos,numero_de_puntos_Pm_gen+i,4);
        A5=q(cont_casos,numero_de_puntos_Pm_gen+i,5);
        A6=q(cont_casos,numero_de_puntos_Pm_gen+i,6);
        J_o(2,:)=subs(Jr_con_delta(2,:));

        E2=casos(cont_casos,2); E1=casos(cont_casos,1);
        A1=q(cont_casos,2*numero_de_puntos_Pm_gen+i,1);
        A2=q(cont_casos,2*numero_de_puntos_Pm_gen+i,2);
        A3=q(cont_casos,2*numero_de_puntos_Pm_gen+i,3);
        A4=q(cont_casos,2*numero_de_puntos_Pm_gen+i,4);
        A5=q(cont_casos,2*numero_de_puntos_Pm_gen+i,5);
        A6=q(cont_casos,2*numero_de_puntos_Pm_gen+i,6);
        J_o(3,:)=subs(Jr_con_delta(3,:));

        for j=0:(num_filas_Jr_con_delta-1)
            W(cont_casos*num_filas_Jr_con_delta*i-
            j,:)=J_o(num_filas_Jr_con_delta-j,:); % matriz de observacion
        end
    end
end

% Pseudo-inversa de la matriz de observacion
Pseinv_W=(inv(W'*W))*W';

% correcciones para el DH: delta_beta=[delta_al delta_theta1]'
delta_beta=Pseinv_W*delta_P;

% Vector de correcciones calculadas en cada iteracion (debe converger a 0)
V_correc_tras_cada_iteracion(:,iter)=delta_beta;

```

```

% Vector de correcciones acumuladas tras un numero 'iter' de iteraciones
% (debe converger a los errores supuestos antes)

delta_d1=delta_d1+delta_beta(1);
delta_alphal=delta_alphal+delta_beta(2);
delta_a1=delta_a1+delta_beta(3);
delta_theta2=delta_theta2+delta_beta(4);
delta_E2=delta_E2+delta_beta(5);
delta_E1=delta_E1+delta_beta(6);

V_correc_acumuladas(:,iter)=[delta_d1 delta_alphal delta_a1 delta_theta2
delta_E2 delta_E1]';

% Nuevos valores para P_con_delta_beta con las correcciones realizadas al
% modelo.

clear delta_P_filas;
j=1;
for cont_casos=1:size(casos,1)
    for i=1:numero_de_puntos_Pm_gen
        E2=casos(cont_casos,2); E1=casos(cont_casos,1);
        A1=q(cont_casos,i,1); A2=q(cont_casos,i,2);
        A3=q(cont_casos,i,3); A4=q(cont_casos,i,4); A5=q(cont_casos,i,5);
        A6=q(cont_casos,i,6);
        delta_P_filas(j,1)=Ps(cont_casos,i,1)-
        subs(PO_LASER_con_delta_beta(1,4)); % 16 Puntos en Plano YZ (plano
        vertical paralelo al lado largo de la mesa, con lo que el error esta en
        Xbase)

        E2=casos(cont_casos,2); E1=casos(cont_casos,1);
        A1=q(cont_casos,numero_de_puntos_Pm_gen+i,1);
        A2=q(cont_casos,numero_de_puntos_Pm_gen+i,2);
        A3=q(cont_casos,numero_de_puntos_Pm_gen+i,3);
        A4=q(cont_casos,numero_de_puntos_Pm_gen+i,4);
        A5=q(cont_casos,numero_de_puntos_Pm_gen+i,5);
        A6=q(cont_casos,numero_de_puntos_Pm_gen+i,6);
        delta_P_filas(j,2)=Ps(cont_casos,numero_de_puntos_Pm_gen+i,2)-
        subs(PO_LASER_con_delta_beta(2,4)); % 16 Puntos en Plano XZ (plano
        vertical paralelo al lado corto de la mesa, con lo que el error esta en
        Ybase)

        E2=casos(cont_casos,2); E1=casos(cont_casos,1);
        A1=q(cont_casos,2*numero_de_puntos_Pm_gen+i,1);
        A2=q(cont_casos,2*numero_de_puntos_Pm_gen+i,2);
        A3=q(cont_casos,2*numero_de_puntos_Pm_gen+i,3);
        A4=q(cont_casos,2*numero_de_puntos_Pm_gen+i,4);
        A5=q(cont_casos,2*numero_de_puntos_Pm_gen+i,5);
        A6=q(cont_casos,2*numero_de_puntos_Pm_gen+i,6);
        delta_P_filas(j,3)=Ps(cont_casos,2*numero_de_puntos_Pm_gen+i,3)-
        subs(PO_LASER_con_delta_beta(3,4)); % 16 Puntos en Plano XY (superficie
        mesa, con lo que el error está en la cota Zbase)
        j=j+1;
    end
end
end

V_correc_tras_cada_iteracion
V_correc_acumuladas

```

V_errores_posic;

The desired result is achieved after 20 iterations.

```

% V_correc_tras_cada_iteracion =
%
% Columns 1 through 6
%
%      2.5261      -2.4744      -0.8153      0.046677      0.26192      0.22285
%      0.003954      0.0032042      0.0017677      0.00080756      0.00029859      7.0228e-005
%      1.7958      -1.2227      -0.72805      -0.097757      0.092338      0.093987
%      0.0065782      0.0032192      0.0011285      5.4383e-006      -0.00030156      -0.0002715
%      -0.10895      -0.0086977      0.098631      0.059879      0.021651      0.0061652
%      -0.75614      -0.17527      0.49519      0.35682      0.14534      0.03935
%
% Columns 7 through 12
%
%      0.14037      0.07674      0.038006      0.017183      0.0070109      0.0024844
%     -1.1631e-005     -2.9501e-005     -2.5332e-005     -1.6791e-005     -9.7019e-006     -5.0865e-006
%      0.061173      0.034333      0.017549      0.0082141      0.00349      0.0013119
%     -0.00017534     -9.7566e-005     -4.9105e-005     -2.2592e-005     -9.4273e-006     -3.4615e-006
%      0.0014933      0.00021234     -9.9251e-005     -0.00012604     -8.469e-005     -4.4372e-005
%      0.0019563      -0.0075152     -0.0076546     -0.0054512     -0.0032938     -0.0017844
%
% Columns 13 through 18
%
%      0.00067594      5.6418e-005     -9.8746e-005     -0.00010077     -6.8299e-005     -3.8797e-005
%     -2.4554e-006     -1.0927e-006     -4.4303e-007     -1.5814e-007     -4.4887e-008     -5.7397e-009
%      0.00040441      7.2713e-005     -2.4058e-005     -3.7516e-005     -2.8213e-005     -1.6927e-005
%     -1.0202e-006     -1.501e-007      9.0399e-008      1.1376e-007      8.1667e-008      4.7839e-008
%     -1.9357e-005     -6.8299e-006     -1.5418e-006      2.581e-007      6.2839e-007      5.2992e-007
%     -0.00088563     -0.00040512     -0.00016967     -6.3452e-005     -1.9744e-005     -3.8307e-006
%
% Columns 19 through 20
%
%     -1.9584e-005     -8.9244e-006
%      4.5662e-009      5.2604e-009
%     -8.892e-006     -4.2033e-006
%      2.4696e-008      1.1487e-008
%      3.4585e-007      1.9679e-007
%      8.5922e-007      1.5828e-006      (almost zeroes)
%
% V_correc_acumuladas =
%
% Columns 1 through 6
%
%      2.5261      0.051669      -0.76363      -0.71696      -0.45504      -0.23219
%      0.003954      0.0071582      0.0089258      0.0097334      0.010032      0.010102
%      1.7958      0.57305      -0.15499      -0.25275      -0.16041      -0.066425
%      0.0065782      0.0097974      0.010926      0.010931      0.01063      0.010358
%      -0.10895      -0.11765      -0.01902      0.040858      0.062509      0.068674
%      -0.75614      -0.93141      -0.43622      -0.079407      0.065935      0.10529
%
% Columns 7 through 12
%
%     -0.091816     -0.015077      0.02293      0.040113      0.047123      0.049608
%      0.010091      0.010061      0.010036      0.010019      0.010009      0.010004
%     -0.005252      0.029081      0.046629      0.054843      0.058333      0.059645
%      0.010183      0.010085      0.010036      0.010014      0.010004      0.010001
%      0.070168      0.07038      0.070281      0.070155      0.07007      0.070026
%      0.10724      0.099726      0.092072      0.086621      0.083327      0.081542

```

```
% Columns 13 through 18
%
%      0.050284      0.05034      0.050241      0.050141      0.050072      0.050034
%      0.010002      0.010001      0.01      0.01      0.01      0.01
%      0.06005      0.060122      0.060098      0.060061      0.060033      0.060016
%      0.0099998      0.0099996      0.0099997      0.0099998      0.0099999      0.01
%      0.070006      0.069999      0.069998      0.069998      0.069999      0.069999
%      0.080657      0.080252      0.080082      0.080019      0.079999      0.079995
%
% Columns 19 through 20
%
%      0.050014      0.050005      (delta_d1 = 0.05)
%      0.01      0.01      (delta_alpha1 = 0.01)
%      0.060007      0.060003      (delta_a1 = 0.06)
%      0.01      0.01      (delta_theta2 = 0.01)
%      0.07      0.07      (delta_E2 = 0.07)
%      0.079996      0.079997      (delta_E1 = 0.08)
```

A.3. TCL CONCEPTS FOR THE PROGRAMMING OF A POSTPROCESSOR IN THE PLATFORM NX

A.3.1. Introduction. General characteristics of the TCL.

TCL (Tool Command Language) is a scripting language created by John Ousterhout (Berkeley Univ.)¹. It is used for scripted applications, GUIs and testing. Tcl is used on embedded systems platforms, both in its full form and in several other small-footprinted versions. In the NXTM system, it is used to configure the NX/Post.

A TCL script can connect several modules in different programming languages (such as C++), without compiling them again, see Figure A3.1. The main difference between compiled and interpreted languages, like the TCL, arises in the way in which the translation is done (instruction by instruction in this case, while in a compiler the translation is done for the full code after read).

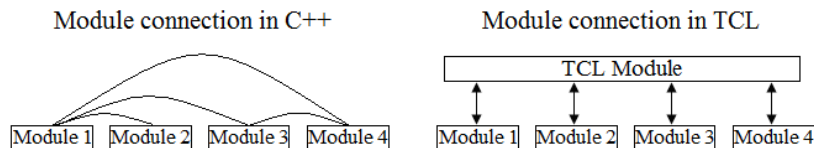


Figure A3.1. Connection amongst modules in TCL and C++

A.3.2. Format of the orders in Tcl

The general syntax of an order in Tcl is of the form:

order options argument 1 argument 2 argument n

The word *order* is the name of the Tcl command or of a procedure Tcl developed previously. The *options* give to the TCL interpreter detailed instructions of the task that the order must develop. The arguments are any type of information that could be processed, changed or used somehow in the execution of the order.

¹ J. K. Ousterhout; "Tcl and the Tk Toolkit", Addison Wesley, 1994, ISBN 020163337X

Other reference books are:

B. B. Welch, K. Jones, J. Hobbs; "Practical programming in Tcl/Tk". ISBN 0130385603

F. Feito, R. J. Segura, F. de Asís; "Programación en Tcl/Tk", Universidad de Jaén, 1997. ISBN 8488942966

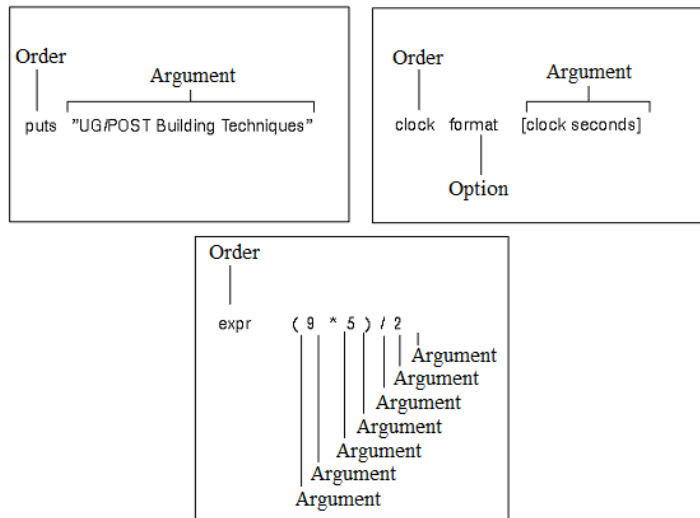


Figure A3. 2. Some examples of orders and arguments.

The order grouping in Tcl, or *script*, is realized by means of a sequence in different lines and between keys. The program comments are preceded by # in order that they are not considered by the TCL interpreter.

```

Script {
  #If the value of X is less than 3, then print
  #the following message: "X is too small"
  if { $X < 3 } {
    ,puts stdout "X is too small"
    ,set X 3
  }
}
Comment

```

Figure A3. 3. Script and comments in TCL

The flow control structures are TCL orders that allow the control of the execution of the program. Like in many other languages, they can be *selective* or *iterative*. The first ones would contemplate conditional structures (*if/elseif*), or multiconditional alternatives like-a-menu (*switch*), whereas the second ones contemplate curls finished by counter (for) and for sentry (*while*).

A.3.3. TCL Variables. Substitution.

The order *set* is used for the assignment of values, without need to initialize them before being used provided that the TCL interpreter creates it simultaneously.

set <variable> <value>

A variable can keep numbers or chains of characters (letters, numbers, symbols or combination of them). Nevertheless, in TCL everything is interpreted as character chains which makes necessary to use the mechanism of *substitution* (by means of the operator \$).

Tcl allows *local* and *global* variables. The local variables are used inside a procedure (*proc*). When the procedure is called, the variable is created, and when it finishes the variable is deleted (*unset <variable>*). A global variable (*global <variable>*) allows to store information amongst calls to different procedures, and has the same identifier independently of its location in the program

A.3.4. Event Handler syntax for the KUKA KRC2 controller

This section analyzes the morphology of the .tcl file adapted to the control KUKA KRC2 for a 6R KR15/2. Basically, the Event Handler has two differentiated parts:

- a) An initialization of variables, in many cases describing the machine or the functioning of the NC. These variables, together with the global variables that NX associates with every event, will be used by the described procedures later.
- b) Una sucesión de procedimientos (*proc*) que, ante un evento a postprocesar, realizan las siguientes tareas:
- c) A succession of procedures (*proc*) that, with an event to postprocess, realize the following tasks:
 - i. Load the variables that are going to be indispensable for the data processing that has to be offered to the controller of the machine tool or robot.
 - ii. Realize the necessary operations with the above mentioned variables.
 - iii. After having all the necessary information in order that the machine tool could materialize the event that wants to be postprocessed, puts all this information to disposition on the

Definition File to be able to write the Output File (with the necessary format, as it will be described later on)

i) Variable definition

A full list (and its explanation) of the variables coming from the generation of the toolpath is available with the installation of the NX™ system (see *mom* variables at {UGII_BASE_DIR} \UGDOC \html_files \ugpost \index.html). These ones and some other user variables are defined as *global* at the beginning of the program, before any *proc* (see code below).

ii) Procedure definition

As explained in the paragraph dedicated to Event Handler's concept, it is necessary to define a procedure (*proc*) for every event to be carried out in the machine. Some of these events come implicitly defined by the way of handling the information of the module of mechanized the platform NX (like the concepts of *start of program*, *start of group* or *start of path*) and necessarily we will find the procedure that will develop the corresponding actions to realize in each of them. Others will be defined by the user according to the characteristics of the machine and of the process, like in the case of the motions in CP tracking (see Section 4.6.3.).

In the most elementary version of *proc* for a linear move, this block of process just sends the information of the destination point and TCP rate to the corresponding templates of the Definition File, that will give KUKA KRL format to the output file as explained later. In this case, it is not necessary to invoke variables since they are *global* and the call is done directly from the Block_Template of the Definition File (see paragraph 1.3.2.).

```
proc MOM_linear_move {} {  
  MOM_do_template Velocidad  
  MOM_do_template Linear  
}
```

Figure A3. 4. Basic processing at the Event Handler of a linear motion.

As an improvement of this processing, and at the expense of the existence of a linear track and a rotary table, it is proposed a review of the previous basic process that solves the adoption of extreme configurations in the robot for remote points of the same one, by means of a *fuzzy*^{2,3} controller, implemented in Matlab⁴.

The call is done by means of the TCL command *catch*, by which the values of EE's position and of the additional external joints (vector *k*) are passed to the executable containing the fuzzy model. The increments obtained for the additional external joints are the last two values of the vector *a*.

```

proc MOM_linear_move {} {
    global mom_pos mom_prev_pos
    global mov_carro mov_mesa
    global k E2 E1 a t

    # Five values passed to the .exe with the IKP and the fuzzy engine.
    set k(0) $E2
    set k(1) $E1
    set k(2) $mom_pos(0)
    set k(3) $mom_pos(1)
    set k(4) $mom_pos(2)

    MOM_output_to_listing_device "k: $k(0) , $k(1) , $k(2) , $k(3) , $k(4)"

    catch {exec C:\WX5\WACH\auxiliary\javi2_matlabfuzz [array get k]} a

    set t(6) [lindex $a 0]
    set t(7) [lindex $a 1]

    set E2 [expr $k(0) + $t(7)]
    set E1 [expr $k(1) + $t(6)]

    set mov_mesa $E2
    set mov_carro $E1

    MOM_do_template Velocidad
    MOM_do_template Linear
}

```

Figure A3. 5. Event Handler processing of a linear motion relocating the external joints.

² J. Andres, L. Gracia, J.Tornero; "Inverse kinematics of a redundant manipulator for CAM integration. An industrial perspective of implementation", ICM09.

³ J. Andres, L. Gracia, J.Tornero; "TOOLPATH POSTPROCESSING FOR THREE AXES MILLING IN REDUNDANT ROBOTIC WORKCELLS BY MEANS OF FUZZY INTEGRATION IN A CAM PLATFORM", ICM09.

⁴ J.-S. Roger Jang, N. Gulley; "Fuzzy Logic Toolbox: User's Guide"; Revised for Version 2.2.7 (Release 2008a), The MathWorks, Inc. 2008

Finally and after all these calculations in which the obtained values are kept as new definite variables, all the information is at the disposal of the Definition File: *MOM_do_template Linear*

iii) Transcription of the Event Handler of the NX-KUKA KRC2postprocessor

```
#####
# CN:          KR15/2 - KRC2 KUKA (_v3)
# Revisiones  20-3-2010 # Javier Andres #
#####

# Global vble def
# Machine Kinematic
set mom_kin_machine_resolution      0.001
set mom_kin_arc_output_mode        FULL_CIRCLE
## Defines how circles will be output by the post. Only circles generated
in the operation can be output as circles. LINEAR will output linear
moves based on the tolerances defined on the arc in the operation.
QUADRANT will output circles only on quadrant boundaries (divide arcs en
cuadrantes). FULL_CIRCLE will output arcs up to 360 degrees (Kuka
controller stands for full_circle)
set mom_kin_helical_arc_output_mode LINEAR
## linearizes helix motions, not allowed at KRC2
set mom_kin_arc_valid_plane        ANY
set mom_kin_min_arc_length         0.01
set mom_kin_min_arc_radius         0.1
set mom_kin_max_arc_radius         5000
# below or above these values, it inearizes.
set mom_kin_machine_type           5_axis_dual_head
set mom_kin_4th_axis_direction     "MAGNITUDE_DETERMINES_DIRECTION"

#set mom_kin_4th_axis_plane        "ZX"
set mom_kin_4th_axis_leader        "B"
set mom_kin_4th_axis_rotation      "standard"
set mom_kin_4th_axis_type          "Head"
set mom_kin_5th_axis_direction     "MAGNITUDE_DETERMINES_DIRECTION"

#set mom_kin_5th_axis_plane        "YZ"
set mom_kin_5th_axis_leader        "C"
set mom_kin_5th_axis_rotation      "standard"
set mom_kin_5th_axis_type          "Head"
set mom_kin_rapid_feed_rate        12000
set mom_kin_tool_change_time       30.0
set mom_sys_spindle_direction_code(OFF) "FALSE"
set mom_sys_output_file_suffix     ".src"

set pto_sim(0)    0
set pto_sim(1)    0
set pto_sim(2)    0
set pto_per(0)    0
set pto_per(1)    0
set pto_per(2)    0
```

```

set angulo_arco_circ      0
set abs_ang_giro         0
set R(0)                 0
set R(1)                 0
set R(2)                 0
set mod_R                0
set Vp(0)                0
set Vp(1)                0
set Vp(2)                0
set mod_Vp               0
set div_mod              0
set Rp(0)                0
set Rp(1)                0
set Rp(2)                0
set num_despl            0
set despl                200
set sentido              1
set cero_carro           -2100
set mov_carro            0
set mov_mesa             0
set k(0) 0
set k(1) 0
set k(2) 0
set k(3) 0
set k(4) 0
# HOME:
set E2 45
# set E1 -2977.17 E2 0
set E1 -2500
set a "void"
set t(0) 0
set t(1) 90
set t(2) 0
set t(3) -90
set t(4) 90
set t(5) 0
set t(6) 0
set t(7) 0

#####
proc MOM_start_of_program {} {}
#####
proc MOM_start_of_group {} {
    global mom_group_name mom_parent_group_name
    if { $mom_group_name == $mom_parent_group_name } {

        MOM_output_to_listing_device
            "PROGRAM: $mom_parent_group_name\n"
        MOM_output_text "&ACCESS RVP"
        MOM_output_literal "&PARAM" TEMPLATE =
C:\\KRC\\Roboter\\Template\\ExpertVorgabe"
        MOM_output_text "&PARAM EDITMASK = *"
        MOM_output_text "DEF $mom_parent_group_name ()"
        MOM_output_text "BAS (#INITMOV,0)"
        MOM_output_text "PTP XHOME"
        MOM_output_text "\\$APO.CVEL = 100"
    } else {
        MOM_output_to_listing_device

```

```

                                "SUB-PROG: $mom_group_name"
    }
}

#####
proc MOM_machine_mode { } { }
#####
proc MOM_start_of_path { } {
    global mom_path_name mom_parent_group_name mom_group_name
mom_fixture_offset_value
    global mom_csys_matrix
    if {[info exists mom_parent_group_name]} {
        MOM_output_to_listing_device
            "OPERATION: $mom_path_name"
    } else {
        set mom_parent_group_name $mom_path_name
        set mom_group_name $mom_path_name
        MOM_start_of_group
    }
    if { $mom_fixture_offset_value == 0 } {
        set mom_fixture_offset_value 1
    }
        MOM_do_template base
        MOM_do_template tool
}

#####
proc MOM_set_csys { } {
    global mom_csys_matrix
}
#####
proc MOM_msys { } {
    global mom_msys_matrix mom_msys_origin
}

#####
proc MOM_first_tool {} {
    global mom_tool_number
}
#####
proc MOM_tool_change {} {
    global mom_tool_number mom_next_tool_number mom_next_tool_status
        MOM_output_text "PTP HOME"
        MOM_first_tool
        MOM_output_text "PTP HOME"
}
#####
proc MOM_rapid_move {} {
    MOM_linear_move
}
#####
proc MOM_linear_move {} {
    global mom_pos mom_prev_pos
    global mov_carro mov_mesa
    global k E2 E1 a t
# Asignation of the 5 values passed to the IKP.exe
    set k(0) $E2

```

```

        set k(1) $E1
        set k(2) $mom_pos(0)
        set k(3) $mom_pos(1)
        set k(4) $mom_pos(2)
        MOM_output_to_listing_device "k: $k(0) , $k(1) , $k(2) , $k(3) ,
$k(4)"

# given k, .exe gives back the increments for E1 & E2 within the chain a.
    catch {exec C:\\NX5\\MACH\\auxiliary\\javi2_matlabfuzz [array get
k]} a

        set t(6) [lindex $a 0]
        set t(7) [lindex $a 1]

# new ext. Joint values
    set E2 [expr $k(0) + $t(7)]
    set E1 [expr $k(1) + $t(6)]

        set mov_mesa $E2
        set mov_carro $E1

        MOM_do_template Velocidad
        MOM_do_template Linear
    }

#####
proc MOM_circular_move {} {
global mom_arc_direction mom_arc_angle mom_pos_arc_center mom_prev_pos
mom_pos mom_arc_radius
global gb_sim gb_per angulo_arco_circ pruebatan
global R mod_R pto_sim Vp mod_Vp div_mod Rp pto_per angulo_arco_circ
global mom_pos_arc_plane mom_out_angle_pos mom_tool_axis mom_pos_arc_axis

        MOM_do_template Velocidad

# Calculo del Radio entre en Punto Inicial y el Centro del Arco, y de su
modulo
    set R(0) [expr $mom_pos_arc_center(0) - $mom_prev_pos(0)]
        set R(1) [expr $mom_pos_arc_center(1) - $mom_prev_pos(1)]
    set R(2) [expr $mom_pos_arc_center(2) - $mom_prev_pos(2)]
    set mod_R [expr sqrt(pow($R(0),2)+pow($R(1),2)+pow($R(2),2))]

# a) Cálculo del Punto Simétrico del Punto Inicial
    set pto_sim(0) [expr 2*$mom_pos_arc_center(0) - $mom_prev_pos(0)]
        set pto_sim(1) [expr 2*$mom_pos_arc_center(1) - $mom_prev_pos(1)]
    set pto_sim(2) [expr 2*$mom_pos_arc_center(2) - $mom_prev_pos(2)]

# b) Cálculo del Punto Perpendicular.
# b.1.) Calculo de un vector perpendicular (Vp) a R y al vector normal al
plano del arco (mom_pos_arc_axis), y de su modulo
    set Vp(0) [expr $R(1)*(-$mom_pos_arc_axis(2)) - $R(2)*(-
$mom_pos_arc_axis(1))]
        set Vp(1) [expr $R(2)*(-$mom_pos_arc_axis(0)) - $R(0)*(-
$mom_pos_arc_axis(2))]
    set Vp(2) [expr $R(0)*(-$mom_pos_arc_axis(1)) - $R(1)*(-
$mom_pos_arc_axis(0))]
    set mod_Vp [expr sqrt(pow($Vp(0),2)+pow($Vp(1),2)+pow($Vp(2),2))]

```

```

# b.2) Calculo del vector Rp (perpendicular a R y en el plano del arco), y
del mismo modulo que R
    set div_mod [expr $mod_R / $mod_Vp]
    set Rp(0) [expr $Vp(0)*$div_mod]
    set Rp(1) [expr $Vp(1)*$div_mod]
    set Rp(2) [expr $Vp(2)*$div_mod]

# b.3) Calculo del punto auxiliar (pto_per) para el comando CIRC en KRL
    set pto_per(0) [expr $mom_pos_arc_center(0)+$Rp(0)]
    set pto_per(1) [expr $mom_pos_arc_center(1)+$Rp(1)]
    set pto_per(2) [expr $mom_pos_arc_center(2)+$Rp(2)]

# c) Angulo
    set abs_ang_giro [expr abs($mom_arc_angle)]
    set angulo_arco_circ [expr $abs_ang_giro]
    MOM_do_template Circular
}
#####
proc MOM_end_of_path { } {
    proc hiset { v1 } {
        upvar $v1 v2
        if { [info exists v2] } { return 1 } else { return 0 }
    }
}
#####
proc MOM_end_of_group { } { }
#####
proc MOM_end_of_program {} {
    global mom_parent_group_name mom_sys_output_file_suffix
    mom_output_file_full_name mom_output_file_directory
    MOM_output_text "PTP HOME"
    MOM_output_text "END"
    MOM_close_output_file $mom_output_file_full_name

    set new_file
    $mom_output_file_directory$mom_parent_group_name$mom_sys_output_file_suffi
x

    file rename -force $mom_output_file_full_name $new_file

    MOM_output_to_listing_device "\nFichero de mecanizado: $new_file\n"
}
#####
proc MOM_catch_warning { } {
    global mom_warning_info

    if { $mom_warning_info == "WARNING: ONE AXIS ARC MOVE; ABORTED TO LINEAR
MOVE" } {

        # IGNORE

    } else {
        MOM_output_to_listing_device " * ERROR * $mom_warning_info *"
    }
}
#####
# END OF PROGRAM
#####

```

A.3.5. Definition File syntax for the KUKA KRC2 controller

The Definition File, as a TCL extension of the Event Handler, mainly contains static information about a specific machine tool or robot so that the Post could give exit format towards the Output File of the event that is postprocessed, namely: general attributes of the machine (*FORMAT*), addresses of commands supported by the controller (*ADDRESS*) with the attributes of every address (format, max, min) and a set of templates (*BLOCK_TEMPLATE*) that describe how the addresses of previous commands conjugate to shape a specific action in the robot.

In the following, we revise the most significant lines before the transcription of the complete code of the Definition File

i) *General attributes of the machine (FORMATTING)*

The first lines of a definition file are as follows:

```

FORMATTING
{
    WORD_SEPARATOR    " "
    END_OF_LINE      ""

    FORMAT Block_num "&_4_00"
    FORMAT Coordinate "&_4.30"
    FORMAT Socket    "%02d"
    FORMAT Feed      "&_4_00"
    FORMAT Str        "%s"
    FORMAT Zero_int  "&_01_0_"

```

Figure A3. 6. First lines of a definition file

- **WORD_SEPARATOR " "**
It forces to the Postprocessor to insert a chain of characters (in this case a space) between all the other chains of the ADDRESS overturned in the Output File.
- **END_OF_LINE ""**
It forces to the Postprocessor to place a chain of characters (none in this case) at the end of other chains of the ADDRESS overturned in the Output File.
- **FORMAT <name> &abcdef**

It defines data formats (decimal, integer, chain of characters) to the type of datum <name>. The analysis of the characters after the name is the following:

- a = + ó _ (+ forces + at positive values, _ do not)
- b = 0 ó _ (0, zeroes fulfil the digits reserved for the integer part, _ do not)
- c = {0, 1, 2, ..., 9} (number of digits of the integer part)
- d = . or _ ('.' Forces the decimal point, _ do not)
- e = {0,1,2, ..., 9} (number of digits of the decimal part)
- f = 0 or _ (0, zeroes fulfil the digits reserved for the decimal part, _ do not)
- FORMAT Str "%s", for chains of characters

ii) ADDRESS

Following lines of the Definition File establish the addresses that will be used in the Output File, as they could be the addresses X, Y, or Z for the position coordinates of the TCP. With the syntaxes that shown below there are defined the attributes of the above mentioned addresses:

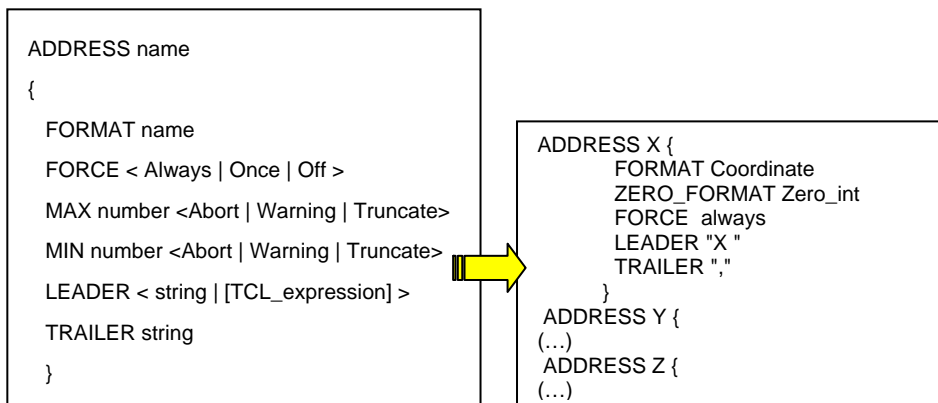


Figure A3. 7. Address definition in the .def code. Right, addresses to command the X position of the TCP.

- ADDRESS name:
It is the name of the address

- **FORMAT name:**
Instructs NX/Post to use format name to print the value of the address expression to the output.
- **FORCE:**
Always (Instructs Post to always output the value of an address expression)/ Once (Instructs Post to output the value of the next address expression, (Default)) / Off (Instructs Post to not force the output of the value of an address expression. Instead, output the value of an address expression if and only if the value is different than the previous address expression value for this address.)
- **MAX number :**
Instructs Post to use number as the maximum value to be output by this address. The options are: Abort, if Max is violated then Abort creation of NC file; Warning, if Max is violated then write a warning message to the system log file and continue; Truncate, if Max is violated then write a warning message to the system log and continue using the Max value as the output value of the address.
- **MIN number:**
Instructs Post to use number as the minimum value to be output by this address. The options are: Abort, if Min is violated then Abort creation of NC file; Warning, if Min is violated then write a warning message to the system log file and continue; Truncate, if Min is violated then write a warning message to the system log and continue using the Min value as the output value of the address.
- **LEADER string:**
Precede this address with *string* in every template in which the address appears. Default is the Address name.
- **TRAILER string:**
Follow this address with *string* in every template in which the address appears. Default is the empty string.

iii) ***BLOCK_TEMPLATE***

Finally the addresses with the formats of the beginning are inserted in the template that gives the definitive form to the proper commands of the language KRL in the Output File. The above mentioned form is given

definitively in the successive *Block Templates* that are called from the Event Handler (by means of the procedure *MOM_do_template*). The structure of a Block template is the following:

```
BLOCK_TEMPLATE name
{
  Address_Name [Address_Expression]
  \nows
  \opt
  \ldr=string1
  \trlr=string
  "string"
}
```

Figure A3. 8. Structure of a Block template.

- **Address_Name**
The name of a previously defined Address. The attributes of that Address determine the format of the output for this Block Template member.
- **Address_Expression (ae)**
A TCL expression whose value should be sent to the output. This expression can be any valid TCL expression. Post will ask TCL to evaluate this expression and then output its value using the format indicated by the Address Format attribute.
- **\nows**
An optional switch. If present then the value is not followed by the WORD_SEPARATOR in the generated output. If not present then the value is followed by the WORD_SEPARATOR in the generated output.
- **\opt**
An optional switch. If present and the ae can not be evaluated then ignore this address. If it is not present and the ae cannot be evaluated an error is issued.
- **\ldr=string1**

An optional switch. If present then precede this address with `string1` in this Block Template (not in every template in which the address appears; use LEADER in the address definition for that).

- `\trlr=string1`
An optional switch. If present then follow this address with `string1` in this Block Template (not in every template in which the address appears; use TRAILER in the address definition for that).
- `"string"`
A string to output literally. This string follows the octal rule. This string may also be qualified with switches.

There can be any number of *Address* members and any number of *string* members in a Block Template. They may appear in any order.

```

BLOCK_TEMPLATE Linear {
  "LIN {E6POS:"
  X[$mom_pos(0)]
  Y[$mom_pos(1)]
  Z[$mom_pos(2)]
  "A 0,"
  A1[$mom_out_angle_pos(0)]
  A2[$mom_out_angle_pos(1)]
  E1[$mov_carro]
  E2[$mov_mesa]\nows
  "} C_VEL"
}

BLOCK_TEMPLATE Circular {
  "CIRC {E6POS:"
  X[$pto_per(0)]
  Y[$pto_per(1)]
  Z[$pto_per(2)]
  "A 0,"
  A1[$mom_out_angle_pos(0)]
  A2[$mom_out_angle_pos(1)]
  E1[$mov_carro]
  E2[$mov_mesa]\nows
  "}, {E6POS: "
  X[$pto_sim(0)]
  Y[$pto_sim(1)]
  Z[$pto_sim(2)]
  "A 0,"
  A1[$mom_out_angle_pos(0)]
  A2[$mom_out_angle_pos(1)]
  E1[$mov_carro]
  E2[$mov_mesa]\nows
  "}, "
  CA[$angulo_arco_circ]
  "C_VEL"
}

```

Figure A3. 9. Block Templates that give exit format to the *circular* and *linear* motions.

*iv) Transcription of the Definition File of the NX-KUKA
KRC2postprocessor*

```

#####
# CN:          KR15/2 - KRC2
# 20-3-2010 # JAVIER ANDRES #
#####
MACHINE GENERIC_MACHINE
FORMATTING
{
  ORD_SEPARATOR      " "
  END_OF_LINE        " "
  FORMAT BLOCK_NUM  "&__4_00"
}

```

```

FORMAT COORDINATE "&_4.30"
FORMAT SOCKET      "%02D"
FORMAT FEED        "&_4_00"
FORMAT STR         "%S"
FORMAT ZERO_INT    "&_01_0_"
#####
# ADDRESSES
#####
ADDRESS STR {
    FORMAT STR
    FORCE ALWAYS
    LEADER ""
}
ADDRESS X {
    FORMAT COORDINATE
    ZERO_FORMAT ZERO_INT
    FORCE ALWAYS
    LEADER "X "
    TRAILER ", "
}
ADDRESS Y {
    FORMAT COORDINATE
    ZERO_FORMAT ZERO_INT
    FORCE ALWAYS
    LEADER "Y "
    TRAILER ", "
}
ADDRESS Z {
    FORMAT COORDINATE
    ZERO_FORMAT ZERO_INT
    FORCE ALWAYS
    LEADER "Z "
    TRAILER ", "
}
ADDRESS CA {
    FORMAT COORDINATE
    ZERO_FORMAT ZERO_INT
    FORCE ALWAYS
    LEADER "CA "
    TRAILER ""
}
ADDRESS A0 {
    FORMAT COORDINATE
    ZERO_FORMAT ZERO_INT
    FORCE ALWAYS
    LEADER "A "
    TRAILER ", "
}
ADDRESS A1 {
    FORMAT COORDINATE
    ZERO_FORMAT ZERO_INT
    FORCE ALWAYS
    LEADER [ $MOM_KIN_4TH_AXIS_LEADER ]
    TRAILER ", "
}
ADDRESS A2 {
    FORMAT COORDINATE
    ZERO_FORMAT ZERO_INT
    FORCE ALWAYS
    LEADER [ $MOM_KIN_5TH_AXIS_LEADER ]
    TRAILER ", "
}
ADDRESS E1 {
    FORMAT COORDINATE
    ZERO_FORMAT ZERO_INT

```

```

        FORCE ALWAYS
        LEADER "E1 "
            TRAILER ", "          }
ADDRESS E2 {
        FORMAT COORDINATE
        ZERO_FORMAT ZERO_INT
        FORCE ALWAYS
        LEADER "E2 "          }
ADDRESS M_SPIN {
        FORMAT SOCKET
        FORCE OFF
        LEADER "$OUT["
            TRAILER "]"          }
ADDRESS VEL {
        FORMAT COORDINATE
        ZERO_FORMAT ZERO_INT
        FORCE OFF
        LEADER "$VEL.CP="      }
ADDRESS B {
        FORMAT BLOCK_NUM
        MAX 999999 TRUNCATE
        MIN 1
        FORCE OFF
        LEADER "$BASE = BASE_DATA["
            TRAILER "]"        }
ADDRESS T {
        FORMAT BLOCK_NUM
        MAX 999999 TRUNCATE
        MIN 1
        FORCE OFF
        LEADER "$TOOL = TOOL_DATA["
            TRAILER "]"        }

#####
# BLOCK_TEMPLATES
#####
BLOCK_TEMPLATE BASE {
    B[$MOM_FIXTURE_OFFSET_VALUE]
}

BLOCK_TEMPLATE TOOL {
    T[$MOM_TOOL_NUMBER]
}

BLOCK_TEMPLATE LINEAR {
    "LIN {E6POS:"
    X[$MOM_POS(0)]
    Y[$MOM_POS(1)]
    Z[$MOM_POS(2)]
    A0[-$MOV_MESA]
    A1[$MOM_OUT_ANGLE_POS(0)]
    A2[$MOM_OUT_ANGLE_POS(1)]
    E1[$MOV_CARRO]
    E2[$MOV_MESA]\NOWS
    " } C_VEL"          }
# See KUKA Expert programming manual for C_VEL information

BLOCK_TEMPLATE CIRCULAR {

```

```
"CIRC {E6POS:"
X[$PTO_PER(0)]
Y[$PTO_PER(1)]
Z[$PTO_PER(2)]
A0[-$MOV_MESA]
A1[$MOM_OUT_ANGLE_POS(0)]
A2[$MOM_OUT_ANGLE_POS(1)]
E1[$MOV_CARRO]
E2[$MOV_MESA]\NOWS
"}, {E6POS: "
X[$PTO_SIM(0)]
Y[$PTO_SIM(1)]
Z[$PTO_SIM(2)]
A0[-$MOV_MESA]
A1[$MOM_OUT_ANGLE_POS(0)]
A2[$MOM_OUT_ANGLE_POS(1)]
E1[$MOV_CARRO]
E2[$MOV_MESA]\NOWS
"}, "
CA[$ANGULO_ARCO_CIRC]
"C_VEL"
}

BLOCK_TEMPLATE VELOCIDAD {
    VEL[$MOM_FEED_RATE/60000]
}

BLOCK_TEMPLATE SPINDLE_OFF {
    M_SPIN[$MOM_SYS_SPINDLE_DIRECTION_CODE(OFF)] }
}

#####
# END OF PROGRAM
#####
```

A.4. CHARACTERISTIC LENGTH L OF THE KUKA KR-15/2 (CORRESPONDING TO SECTION 2.4.6.)

```

clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
KUKA_SDH_solo= [-pi/2   .300   0   .675   0;
               0   .650   0   0   0;
               pi/2   .155   0   0   0;
               pi/2   0   0   -0.600   0;
               pi/2   0   0   0   0;
               0   0   0   -0.140   0];

% TO PLOT THE ROBOT WITH CORKE'S PLOT
L1_SDH_solo=link([-pi/2   .300   0   .675   0], 'standard');
L2_SDH_solo=link([0   .650   0   0   0], 'standard');
L3_SDH_solo=link([pi/2   .155   0   0   0], 'standard');
L4_SDH_solo=link([pi/2   0   0   -0.600   0], 'standard');
L5_SDH_solo=link([pi/2   0   0   0   0], 'standard');
L6_SDH_solo=link([0   0   0   -0.140   0], 'standard');
ROB_KUKA_SDH_solo=robot({L1_SDH_solo L2_SDH_solo L3_SDH_solo L4_SDH_solo
L5_SDH_solo L6_SDH_solo});
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% drivebot(ROB_KUKA_SDH_solo)
% q_SDH_solo=[0 -pi/2 0 0 pi/2 0];
% T_SDH_solo=fkine(ROB_KUKA_SDH_solo,q_SDH_solo);

% from the DH parameters
aM=max(abs(KUKA_SDH_solo(:,2)));
bM=max(abs(KUKA_SDH_solo(2:end,4)));
M=max(aM,bM);

% thus, with M we calculate the non-dimensional DH params
KUKA_non_dims=[KUKA_SDH_solo(:,1) KUKA_SDH_solo(:,2)/M
KUKA_SDH_solo(:,3) KUKA_SDH_solo(:,4)/M KUKA_SDH_solo(:,5)]; %we dont mind
the value of b1

% So now, i'm implementing the search of eq. (35) of Khan and Angeles 2006
% After defining my fuction kF2, lets do
init_guess=[1.5 -pi/2 pi/2 0 pi/2]; % of [Mraya theta2 theta3 theta4
theta5], dont miss it
[x,FVAL]=fminsearch(@(x) kF2(x,KUKA_non_dims),init_guess);

% x = 1.8543   -1.9280   1.0530   0.0000   1.8299
% o lo que es lo mismo, pasando los angulos a grados
% x = (Mraya=1.8543) [-110.4667   60.3314   0.0002  104.8436]
Mraya=x(1);
q=[0 x(2:end) 0] % theta6 was not included, as it does not affect the
condition number for this particular architecture.

figure(1)
plot(ROB_KUKA_SDH_solo,q);

% The characteristic length is thus computed as
L=M/Mraya % L=0.3506 m (el RSW4 da .350572 m) OK!!!
kF=sqrt(FVAL)
KCI=inv(kF)*100;

```



```

% the KCI of this manipulator can still be improved dramatically by noting
% that the condition number is highly dependent on the location of the
operation
% point of the end-effector. The robot DH parameters given in Table 5.2 do
not
% account for the geometry of the EE.

% Finally, rms of distances "ro" from 1 to n of the OP to the n axes of
the R joints

    n=size(KUKA_non_dimens,1);
    [G,T,ej]=dirkin(q, KUKA_non_dimens);

    sumatorio=0;
    for i = 1:n
        ro_i=norm((G(:,end-1) - G(:,i)));
        sumatorio=sumatorio+ro_i^2;
    end

rms_ro_raya=sqrt(1/n*(sumatorio)); % homog
rms_ro=L*rms_ro_raya; % en mm

```

A.5. MATLAB CODE FOR THE POSTPROCESSING OF CLSF FOUNDED ON THE VJM WITH PERIODIC RE-EVALUATION.

A.5.1. Code base (founded on algorithms 6.2 and 6.11)

```

close all
clear all; clc
%%% CARGO DATOS %%%%%%%%%%%%%%%
Definicion_modelos_DHidf_VJMidf
T_brida_TCP=eye(4); T_brida_TCP_MDH=eye(4);
Definicion_trayectoria_espiesferica_para_el_estudio_desplazada_2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% a) initial guess of joint position
theta=[pi 0 pi -pi/2 0 0 +pi/2 0 0]; % El HOME de simple
figure(1)
AZ = -37.5; EL = 10; VIEW(AZ,EL); set(gca,'DataAspectRatio',[2 2 2])
plot(KUKA_SDH_todo, theta);
% drivebot(KUKA_SDH_todo)
hold on
for i=1:size(T_CAM,3)
    point_TCP=[T_CAM(1,4,i),T_CAM(2,4,i),T_CAM(3,4,i)];
    plot3(point_TCP(1),point_TCP(2),point_TCP(3),'+'); % TCP position
    point_HEADtool=[T_CAM(1,4,i)+T_CAM(1,3,i),T_CAM(2,4,i)+T_CAM(2,3,i)
    ,T_CAM(3,4,i)+T_CAM(3,3,i)]; % tool head
    line_tool=[[point_TCP];[point_HEADtool]];
    plot3(line_tool(:,1),line_tool(:,2),line_tool(:,3),'r'); % tool
    vector
end
hold off

figure(2)
AXIS ([-1 1 -1 1 -1 1])

```

```

AZ = -37.5; EL = 10; VIEW(AZ,EL); %set(gca,'DataAspectRatio',[2 2 2])
clear time q_plot
cont=0;

theta_threshold_ref=zeros(1,9);
for i=1:(pose-1)
cont=cont+1;
h=gca;
cla(h);
% b) desired pose of the EE
Td=T_CAM(:,:,i);
pd=Td(1:3,4);
Qd=Td(1:3,1:3);
incr_theta=ones(1,8); % para que entre en el while
theta_previo=theta;

while norm(incr_theta,inf)>0.2 % uso la norma infinito (Chebychev)

% c) {p Q T}<--DKP(theta)
[G,T_brida,ei]=dirkin(theta, KUKAstd_todo);
T=T_brida*T_brida_TCP;
Q=T(1:3,1:3); % tool orientation!!
p=T(1:3,4); % tool position!!
ei(:,end)=T(1:3,3); % ei_tool = ztool!!
G(:,end)=T(1:3,4);

% d) incrQ <-- Q' * Qd
incrQ=Q'*Qd;

% e) incrp <-- pd-p
incrp=pd-p;

% f) incrt
Vector_incrQ=1/2*[incrQ(3,2)-incrQ(2,3);incrQ(1,3)-
incrQ(3,1);incrQ(2,1)-incrQ(1,2)];
incr_t=[Q*Vector_incrQ; incrp];

% g)DKP(theta)
% g.1) ==> saco los vector projectors y el Jacobiano
e=T(1:3,3); % este el el eje de la tool
J_mio=jacobianEE_0(KUKAstd_todo, G, theta);
A=J_mio(1:3,:);
B=J_mio(4:6,:);
J=[A; B];

% ---> para evaluar la posicion que tiene el manipulador 6R
theta_solo=[0 theta(4:8)];
[G_solo,T_brida_solo,ei_solo]=dirkin(theta_solo, KUKAstd_solo);
T_solo=T_brida_solo;
Q_solo=T_solo(1:3,1:3);
p_solo=T_solo(1:3,4);
ei_solo(:,end)=T_solo(1:3,3);
G_solo(:,end)=T_solo(1:3,4);
% J_Kf_to_decide=jacobianEE_0(KUKAstd_solo, G_solo, theta_solo); %
tienen que dar idem, efectivamente.
J_Kf_to_decide=jacobianTCP_0(KUKAstd_solo, G_solo, ei_solo, T_solo);
A_Kf_to_decide=J_Kf_to_decide(1:3,:);
B_Kf_to_decide=J_Kf_to_decide(4:6,:);

```

```

    BH_Kf_to_decide=(1/L_solo)*B_Kf_to_decide; % L_solo es la long caract
    H_Kf_to_decide=[A_Kf_to_decide; BH_Kf_to_decide];
    Kf_to_decide=kF(H_Kf_to_decide(:,1:6)); % el joint virtual no lo pongo
en el jacobiano, para que coincida el valor con v12b4 de Huo!
    inv_Kf_to_decide=1/Kf_to_decide;
    % ----> fin evaluacion posicion
    inv_Kf_to_decide_threshold=0.5; % umbral

% h) algoritmo de resolucion VIRTUAL JOINT METHOD (dara incr_theta)
% h.1) la pseudoinv de J y de J_21
weights_fuzzy_psiJ=diag([1 1 1 1 1 1 1 1]);
invWeightsJ=inv(weights_fuzzy_psiJ);
psiJ=J'*inv(J*J');
psiJw=invWeightsJ*J'*inv(J*invWeightsJ*J');

% h.3) aprovechamiento de los grado de libertad
% h.3.1) recolocacion de la cadena lejos de limites articulares
weights_fuzzy_referenceposture=readfis('fuzzy_refposture');
values_to_watch_referenceposture=[theta(5) abs(theta(7)) theta(4)]; %
en el .fis doy valores limites articulares del modelo de SDH que empleo en
el calculo no mecanicos --> theta3=[-74(+16),70(+160)], theta5=[+/-135]
pp=evalfis(values_to_watch_referenceposture,weights_fuzzy_referenceposture
);
weights1=[pp(1) pp(2) 0.01 pp(5) pp(3) 0.02 pp(4) 0.01 0.01] ;
weights1=[pp(1) pp(2) 0.01 0.01 pp(3) 0.01 pp(4) 0.01 0.01] ;
% weights1=[0.1 0.1 0.1 0.3 0.3 0.2 0.4 0.1 0.1]*0.1; % bueno
% weights1=[0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01];
% weights1=[0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1];
theta_ref=[pi 0.5 pi -pi/2 0 0 +pi/2 0 0];
h1=-diag(weights1)*(theta-theta_ref)';

% h.3.2) tendencia a mantener una postura bien condicionada
weights_fuzzy_singularities=readfis('fuzzy_singularities');
values_to_watch_singularities=[theta(5) abs(theta(7))]; % en el .fis
doy valores limites articulares del modelo de SDH que empleo en el calculo
no mecanicos
aa=evalfis(values_to_watch_singularities,weights_fuzzy_singularities);
weights2=[aa(1) aa(2) aa(3) aa(3) aa(3) aa(4) aa(4) aa(4) 0.01];
% weights_fuzzy_singularities=readfis('fuzzy_refposture');
if inv_Kf_to_decide<inv_Kf_to_decide_threshold % quiero 0.4 a 1
    if theta_threshold_ref==zeros(1,9)
        theta_threshold_ref=theta;
    end
    h2=-diag(weights2)*Kf_to_decide*(theta-theta_threshold_ref)';
else
    theta_threshold_ref=zeros(1,9); % si no me paso del umbral kF
deseado o he vuelto dentro de los valores deseados, lo "reseteo"
    h2=zeros(9,1);
end
h=h1+h2;

% h.4) incr_theta que precisamos
% h.4.1) Metodo tradicional
% original
% % incr_theta_orig=psiJw*incr_t+(eye(9)-psiJ*J)*h;
% % incr_theta=incr_theta_orig;
% h.4.2) Metodo AA_98 (basado en algoritmo 6.2 del doc de tesis, da idem
que el tradicional)
% Primer sub-problema

```


A.5.2. Additional sub-functions

The following sub-functions are requested at certain stages in the code described in the previous section:

- Definicion_modelos_DHidf_VJMidf

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CELULA SDH %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
KUKAstdd_todo= [pi/2      .803  0  -.305  0;
                pi/2      0  0  0  1;
                +pi/2     .3  0  -0.675  0;
                0         .65  0  0  0;
                +pi/2     .155 0  0  0;
                +pi/2     0  0  -0.6  0;
                pi/2      0  0  0  0;
                0.3564    0  0  -0.4434 0;
                0         0  0  -0.1197 0]; % del RSW

LM_SDH_todo=link([pi/2 .803 0 -.305 0], 'standard');
LL_SDH_todo=link([pi/2 0 0 0 1], 'standard');
L1_SDH_todo=link([pi/2 .300 0 -.675 0], 'standard');
L2_SDH_todo=link([0 .650 0 0 0], 'standard');
L3_SDH_todo=link([pi/2 .155 0 0 0], 'standard');
L4_SDH_todo=link([pi/2 0 0 -.600 0], 'standard');
L5_SDH_todo=link([pi/2 0 0 0 0], 'standard');
L6_SDH_todo=link([0.3564 0 0 -0.4434 0], 'standard');
L7_SDH_todo=link([0 0 0 -0.1197 0], 'standard');
KUKA_SDH_todo=robot({LM_SDH_todo LL_SDH_todo L1_SDH_todo L2_SDH_todo
L3_SDH_todo L4_SDH_todo L5_SDH_todo L6_SDH_todo L7_SDH_todo});
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L_todo=0.348; % metros

%%% PLOT MDH HEMERO (EN METROS y MDH) %%%%%%%%%
KUKA_MDH_todo=[ pi 0 0 .305 0;
                pi/2 -.803 0 0 1;
                -pi/2 0 0 -.675 0;
                pi/2 .300 0 0 0;
                0 .650 0 0 0;
                pi/2 .155 0 -.600 0;
                -pi/2 0 0 0 0;
                +pi/2 0 0 -.4434 0;
                -0.3564 0 0 -.1197 0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
KUKAstdd_hasta_munyecaca= [pi/2 .803 0 -.305 0;
                           pi/2 0 0 0 1;
                           +pi/2 .3 0 -0.675 0;
                           0 .65 0 0 0;
                           +pi/2 .155 0 0 0;
                           +pi/2 0 0 -0.6 0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
KUKAstdd_solo= [-pi/2 .300 0 .675 0;
                0 .650 0 0 0];
    
```

```

pi/2      .155    0  0      0;
pi/2      0      0 -0.600  0;
pi/2      0      0  0      0;
0.3564    0      0 -0.140  0];

% TO PLOT THE ROBOT WITH CORKE'S PLOT
L1_SDH_solo=link([-pi/2 .300 0 .675 0], 'standard');
L2_SDH_solo=link([0 .650 0 0 0], 'standard');
L3_SDH_solo=link([pi/2 .155 0 0 0], 'standard');
L4_SDH_solo=link([pi/2 0 0 -0.600 0], 'standard');
L5_SDH_solo=link([pi/2 0 0 0 0], 'standard');
L6_SDH_solo=link([0.3564 0 0 -0.140 0], 'standard');
KUKA_SDH_solo=robot({L1_SDH_solo L2_SDH_solo L3_SDH_solo L4_SDH_solo
L5_SDH_solo L6_SDH_solo});
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L_solo=0.3506; % metros

```

- Definicion_trayectoria_espiesferica_para_el_estudio_desplazada
_2

```

t=0; pose=0;
tfin=2*400; % segs
incr_tiempo=5;
sph_center=[0.1; 0.2; 0.25]; % situacion del centro de la trayectoria
esferica
sph_radius=0.15;
for t=0:incr_tiempo:tfin
    pose=pose+1;
    t=t+150; % para que no empiece exactamente en la vertical, donde la mz
de orientacion que defino abajo tendria indeterminacion al no existir r
    p_tool=[sph_radius*cos((2*pi/285)*t)*sin(pi*t/(2*tfin));
sph_radius*sin((2*pi/285)*t)*sin(pi*t/(2*tfin));
sph_radius*cos(pi*t/(2*tfin))] + sph_center;
    % vectores que formaran la matriz de orientacion: [ex_tool ey_tool
ez_tool]
    % tomando la esfera con centro [0 0 0], claramente el vector
de posicion de p_tool es lo que quiero que sea ztool
    ztool=[sph_radius*cos((2*pi/285)*t)*sin(pi*t/(2*tfin));
sph_radius*sin((2*pi/285)*t)*sin(pi*t/(2*tfin));
sph_radius*cos(pi*t/(2*tfin))];
    norma_ztool=norm(ztool);
    ez_tool=ztool/norma_ztool;
    % luego, el ytool será paralelo al plano z=0, y perpendicular
a ztool. Si consirdero el radio r en el plano z=0 haré el producto r x
ztool
    r=[sph_radius*cos((2*pi/285)*t)*sin(pi*t/(2*tfin));
sph_radius*sin((2*pi/285)*t)*sin(pi*t/(2*tfin)); 0];
    ytool=cross(ztool,r);
    norma_ytool=norm(ytool);
    ey_tool=ytool/norma_ytool;
    % Finalmente, perpendicular a estos dos tengo
ex_tool=cross(ey_tool,ez_tool);
    % finalmete, formo la matriz
Q_tool=[ex_tool ey_tool ez_tool];
    % Q_tool=eye(3); % este de aqui tiene una orientacion vertical
T_CAM_i=eye(4); T_CAM_i(1:3,1:3)=Q_tool; T_CAM_i(1:3,4)=p_tool;
T_CAM(:, :, pose)=T_CAM_i;

```

end

- **Calculo_de_Condicionamientos_kF_para_plotear**

```

%%% Condition number achieved after the Method %%%%%%%%%%
theta_solo=[0 theta(4:8)]; % para evaluar la posicion que tiene el
manipulador en sí, sin rail ni mesa
[G_solo,T_brida_solo,ei_solo]=dirkin(theta_solo, KUKAstd_solo);
T_solo=T_brida_solo;
Q_solo=T_solo(1:3,1:3);
p_solo=T_solo(1:3,4);
ei_solo(:,end)=T_solo(1:3,3);
G_solo(:,end)=T_solo(1:3,4); % aprovecho el hueco ultimo de este vector
para meter p, que es de nuevo la brida por no haber tool
J_Kf_to_plot=jacobianEE_0(KUKAstd_solo, G_solo, theta_solo);
A_Kf_to_plot=J_Kf_to_plot(1:3,:);
B_Kf_to_plot=J_Kf_to_plot(4:6,:); BH_Kf_to_plot=(1/L_solo)*B_Kf_to_plot; %
L_solo es dato, junto con los modelos de DH
H_Kf_to_plot=[A_Kf_to_plot; BH_Kf_to_plot];
    inv_w_cond_plot_solo(cont)=1/kF(H_Kf_to_plot);
    clear G_solo T_brida_solo ei_solo theta_solo T_solo Q_solo p_solo
J_Kf_to_plot A_Kf_to_plot B_Kf_to_plot BH_Kf_to_plot H_Kf_to_plot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

- **Analisis_posicionamiento_periorizado_100s**

```

if i==20 | i==40 | i==60 | i==80 | i==100 | i==120 | i==140 | i==160

    thetaold=theta;
    miro_A3=theta(5);
    miro_A5=abs(theta(7));
    fuzzy_table=readfis('fuzzy_movtable');
    ttable=evalfis([miro_A3 miro_A5],fuzzy_table);
    ttable=pi/8;
    sentido_giro=+1;
    thetamesa=sentido_giro*ttable+thetaold(1);
    dtrack=thetaold(2);

new_thetal=IK_KUKA_periorizacion100s(thetamesa,dtrack,thetaold(9),T_CAM(:,
:i));
    theta=[new_thetal thetaold(9)];

    %%% better kF?
    inv_kF_orig=inv_w_cond_plot_solo(cont);
    Calculo_de_Condicionamientos_kF_para_plotear
    inv_kF_despues_fuzzy=inv_w_cond_plot_solo(cont);
    if inv_kF_despues_fuzzy<inv_kF_orig
        sentido_giro=-1;
        thetamesa=sentido_giro*ttable+thetaold(1);
        dtrack=thetaold(2);
new_thetal=IK_KUKA_periorizacion100s(thetamesa,dtrack,thetaold(9),T_CAM(:,
:i));

```

```

        theta=[new_theta1 thetaold(9)];
        Calculo_de_Condicionamentos_kF_para_plotear
        inv_kF_despues_fuzzy=inv_w_cond_plot_solo(cont); %
    end

    %%% repeated?
    inv_kF1=0.35;
    inv_kF2=inv_kF_despues_fuzzy;
    subo_inv_kF=1;
    while inv_kF2>inv_kF1
        subo_inv_kF=subo_inv_kF+1;
        thetamesa=subo_inv_kF*sentido_giro*ttable+thetaold(1);
        dtrack=thetaold(2);
    new_theta1=IK_KUKA_periorizacion100s(thetamesa,dtrack,thetaold(9),T_CAM(:,
    :,i));
        theta=[new_theta1 thetaold(9)];
        inv_kF1=inv_kF2;
        Calculo_de_Condicionamentos_kF_para_plotear
    inv_kF2=inv_w_cond_plot_solo(cont);
    end
    theta;

    if theta(4)<-2.5 | theta(4)>0.4 | theta(5)<-1.3 | theta(5)>1.3 |
    abs(theta(7))<0.7 % comprobar que no he rebasado ningún limite
        theta=thetaold;
    end

    % TRACK
    thetaold2=theta;
    miro_A3=theta(5);
    miro_A5=abs(theta(7));
    fuzzy_table=readfis('fuzzy_movtrack');
    ttrack=evalfis([miro_A3 miro_A5],fuzzy_table);
    ttrack=0.1;
    sentido_despl=-1;
    thetamesa=thetaold2(1);
    dtrack=sentido_despl*ttrack+thetaold2(2);
    new_theta2=IK_KUKA_periorizacion100s(thetamesa,dtrack,thetaold2(9),T_CAM(:,
    :,i));
        theta=[new_theta2 thetaold2(9)];

    %%% better kF?
    inv_kF_orig=inv_kF2;
    Calculo_de_Condicionamentos_kF_para_plotear
    inv_kF_despues_fuzzy=inv_w_cond_plot_solo(cont);
    if inv_kF_despues_fuzzy<inv_kF_orig
        sentido_despl=+1;
        thetamesa=thetaold2(1);
        dtrack=sentido_despl*ttrack+thetaold2(2);
    new_theta2=IK_KUKA_periorizacion100s(thetamesa,dtrack,thetaold2(9),T_CAM(:,
    :,i));
        theta=[new_theta2 thetaold2(9)];
        Calculo_de_Condicionamentos_kF_para_plotear
        inv_kF_despues_fuzzy=inv_w_cond_plot_solo(cont);
    end

    %%% repeated?
    inv_kF1=inv_kF_orig;

```



```

    inv_kF2=inv_kF_despues_fuzzy;
    subo_inv_kF=1;
    while inv_kF2>inv_kF1
        subo_inv_kF=subo_inv_kF+1;
        thetamesa=thetaold2(1);
        dtrack=subo_inv_kF*sentido_despl*ttrack+thetaold2(2);
new_theta2=IK_KUKA_periorizacion100s(thetamesa,dtrack,thetaold2(9),T_CAM(
,:,i));
        theta=[new_theta2 thetaold2(9)];
        inv_kF1=inv_kF2;
        Calculo_de_Condicionamientos_kF_para_plotear
        inv_kF2=inv_w_cond_plot_solo(cont);
    end

    if theta(4)<-2.5 | theta(4)>0.4 | theta(5)<-1.3 | theta(5)>1.3 |
abs(theta(7))<0.7 % comprobar que no he rebasado ningún limite
        theta=thetaold2;
    end
end

```

- Figuras_plots_para_el_estudio_idf

```

figure(5)
for i=1:size(T_CAM,3)
    point_TCP=[T_CAM(1,4,i),T_CAM(2,4,i),T_CAM(3,4,i)];
    plot3(point_TCP(1),point_TCP(2),point_TCP(3),'+'); % TCP position
    %
    point_HEADtool=[T_CAM(1,4,i)+T_CAM(1,3,i),T_CAM(2,4,i)+T_CAM(2,3,i),T_CAM(
3,4,i)+T_CAM(3,3,i)]; % tool head
    point_HEADtool=[T_CAM(1:3,4,i)+0.05*T_CAM(1:3,3,i)']; % tool head
    line_tool=[point_TCP];[point_HEADtool];
    plot3(line_tool(:,1),line_tool(:,2),line_tool(:,3),'r'); % tool vector
end
hold on
%%%%%%%%%%%%%%
% tpeor=175;
%
plotbot_mdh2(KUKA_MDH_todo,q_plotbot(tpeor/5,:), 'fw',T_brida_TCP_MDH,T_CAM
) %
% en tpeor pongo el tiempo en que la kf es mas mala. 5 es el incrttime que
% he usado
% plot(KUKA_SDH_todo, q_plot(tpeor/5,:));
%%%%%%%%%%%%%%
plotbot_mdh2(KUKA_MDH_todo,q_plotbot, 'fw',T_brida_TCP_MDH,T_CAM)
% plot(KUKA_SDH_todo, q_plot);
hold off

%%% EJES MECANICOS, POR SEPARADO

figure(6)
hold on
xlabel('Time t (s)')
ylabel(['MECHANICAL \theta_M (°)'])
plot(time,q_plotbot(:,1)*180/pi, 'r') % la mesa no tiene límite
hold off

```

```

figure(9)
hold on
    xlabel('Time t (s)')
    ylabel(['MECHANICAL d_L (m)'])
    plot(time,q_plotbot(:,2),'g'); % plot(time,0,'g'); plot(time,3,'g');
% rail y sus limites
hold off

figure(7)
hold on
    xlabel('Time t (s)')
    ylabel(['DEGREES MECHANICAL \theta_' num2str(3) '\circ, \theta_'
num2str(4) '\circ, \theta_' num2str(5) '\circ'])
    plot(time,q_plotbot(:,3)*180/pi, 'r'); plot(time,180,'r'); plot(time,-
180,'r'); % A1 y sus limites
    plot(time,q_plotbot(:,4)*180/pi, 'c'); plot(time,25,'c'); plot(time,-
145,'c'); % A2 y sus limites
    plot(time,q_plotbot(:,5)*180/pi, 'g'); plot(time,70,'g'); plot(time,-
210,'g'); % A3 y sus limites
hold off

figure(8)
hold on
    xlabel('Time t (s)')
    ylabel(['DEGREES MECHANICAL \theta_' num2str(6) '\circ, \theta_'
num2str(7) '\circ, \theta_' num2str(8) '\circ'])
    plot(time,q_plotbot(:,6)*180/pi, 'r'); plot(time,360,'r');
plot(time,0,'r'); % A4 y sus limites
    plot(time,q_plotbot(:,7)*180/pi, 'c'); plot(time,495,'c');
plot(time,225,'c'); % A5 y sus limites
    plot(time,q_plotbot(:,8)*180/pi, 'g'); plot(time,360,'g'); plot(time,-
360,'g'); % A6 y sus limites
hold off

figure(4)
hold on
    xlabel('Time t (s)')
    ylabel('1/Kf (Kf=Condition number(Frob))')
    plot(time,inv_w_cond_plot_solo,'g') % la cadena A1-A6 solamente
    % plot(time,inv_w_cond_plot_workcell,'g') % todo el workcell
hold off
    inv_kf_promedio=(sum(inv_w_cond_plot_solo))/(size(time,2))

```

- escribir_ncl

```

fi = fopen('salida.ncl', 'wt');
fprintf(fi, 'UNITS/MM\nMODE/MILL\nLOADTL/1, IN, 0, LENGTH, 0.000000,
OSETNO, 0\n');
fprintf(fi, 'CUTTER/20.000000, 10.000000, 0.000000, 0.000000, 0.000000,
0.000000, 75.000000\n');
fprintf(fi, 'LINTOL/0.030000\n');
fprintf(fi, 'MULTAX/ON\n');
fprintf(fi, 'RAPID/\n');
for k=1:(size(T_CAM,3)-1) % porque el ultimo punto no lo llevo a usar
fprintf(fi, 'GOTO/%f, %f, %f, %f, %f,
%f\n',T_CAM(1,4,k)*1000,T_CAM(2,4,k)*1000,T_CAM(3,4,k)*1000,T_CAM(1,3,k),T
_CAM(2,3,k),T_CAM(3,3,k));

```

```
fprintf(fi, 'EXTAXISTURN / %f\n',q_plotbot(k,1)*180/pi);
fprintf(fi, 'EXTAXISTRACK / %f\n',q_plotbot(k,2)*1000-3000);
end
fprintf(fi, 'MULTAX/OFF \nEND \nFINI \n');
fclose(fi)
open('salida.ncl')
```

