

This work aims to optimize a 6- RUS parallel robot to determine the optimal active joints locations (position and orientation) for a flight simulation task using a PSO (Particle Swarm Optimization) algorithm to escape from local minima and an Interior Point algorithm to accelerate the search for the optimum in the region indicated by PSO i.e. Interior Point algorithm work to find bottom of the valley indicated by PSO.

Advisor: Aníbal Alexandre Campos Bonilla

Joinville, 2015

ANO
2015

CLODOALDO SCHUTEL FURTADO NETO | ORIENTATION WORKSPACE
OPTIMIZATION FOR A 6-RUS PARALLEL ROBOT



UDESC

SANTA CATARINA STATE UNIVERSITY – UDESC
TECHNOLOGICAL SCIENCES CENTER – CCT
POST GRADUATION PROGRAM IN MECHANICAL ENGINEERING– PPGEM

MASTER DISSERTATION

ORIENTATION WORKSPACE OPTIMIZATION FOR A 6-RUS PARALLEL ROBOT

CLODOALDO SCHUTEL FURTADO NETO

JOINVILLE, 2015

Clodoaldo Schutel Furtado Neto

**Orientation Workspace Optimization For a 6-RUS
Parallel Robot**

Dissertation submitted to Santa
Catarina State University as part of the
requirements for obtaining the degree
of Master in Mechanical Engineering.

ADVISOR: Dr. Aníbal Alexandre
Campos Bonilla

Joinville, SC

2015

F992o

Furtado Neto, Clodoaldo Schutel

Orientation workspace optimization for a 6-RUS parallel robot / Clodoaldo Schutel Furtado Neto. – 2015.

256 p. : il. ; 21 cm

Advisor: Aníbal Alexandre Campos Bonilla

Bibliography: p. 123- 127

Dissertation (master) – Santa Catarina State University, Technological Sciences Center, Post Graduation Program in Mechanical Engineering, Joinville, 2015.

1. Parallel Robot 2. Flight Simulator 3. 6-RUS 4. Singularity 5. Optimization
I. Bonilla, Aníbal Alexandre Campos . II. Santa Catarina State University, Post Graduation Program in Mechanical Engineering. III. Title.

CDD 620.1 – 23. ed.

CLODOALDO SCHUTEL FURTADO NETO
OTIMIZAÇÃO DA ORIENTAÇÃO NO ESPAÇO DE TRABALHO
DE UM ROBÔ PARALELO 6-RUS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia Mecânica como requisito parcial para obtenção do título de Mestre em Engenharia Mecânica na área de concentração "Modelagem e Simulação Numérica".

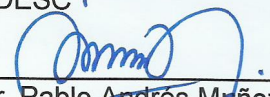
Banca Examinadora

Orientador:

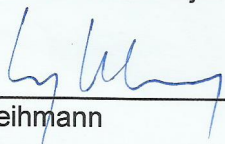


Prof. Dr. Aníbal Alexandre Campos Bonilla
CCT/UDESC

Membros



Prof. Dr. Pablo Andrés Muñoz Rojas
CCT/UDESC



Prof. Dr. Lucas Weihmann
UFSC

Joinville, SC, 28 de julho de 2015.

For my mother Elga.

ACKNOWLEDGEMENTS

I would like to thank the Mechanical Engineering Department of UDESC for accepting me to carry out this master. In a special way express my gratitude.

To Dr. Aníbal Alexandre Campos Bonilla, for guiding me with enthusiasm in this work. I have great affection and appreciation for you, admire as a person and also for your professionalism.

My colleagues and friends Guilherme Espindola, Guilherme Faveri and Rodrigo Trentini, who help me in some stages of this work.

To my mother who taught me my values, had the enormous task of educate my brother and I alone and always sought the best for us.

My brother who was always by my side and made everything in my life more easier.

My dear friend Ana Maria Franco who encouraged me to enter this master, helped me when I faced obstacles and especially for your friendship and affection.

To my dear friend Thiago Kavilhuka who is gone and left many longing. *In memoriam* Thiago Kavilhuka.

To God who gave me health and strength of will to not discourage along the way.

“Hear, I forget. I saw, I remembered. I did, I learned“.

Confucius

ABSTRACT

FURTADO, C. **Orientation Workspace Optimization For a 6-RUS Parallel Robot.** 2015. 256 p. Dissertation (Master in Mechanical Engineering – Area: Design, Analysis and Optimization of Mechanical Systems) – Santa Catarina State University, Post Graduation Program in Mechanical Engineering, Joinville (Brazil), 2015.

The Santa Catarina State University built a flight simulator based on 6-DoF axisymmetric parallel robot which incorporates virtual reality immersion environment. This flight simulator presents 6-RUS kinematic chain which is the second most common architecture for this application. Hunt proposed this chain architecture early in 1983. Parallel robots are closed-loop mechanisms that present good performance in terms of accuracy, rigidity and ability to manipulate large loads. This work aims to optimize a 6- RUS parallel robot to determine the optimal active joints locations (position and orientation) for a flight simulation task using a PSO (Particle Swarm Optimization) algorithm to escape from local minima and an Interior Point algorithm to accelerate the search for the optimum in the region indicated by PSO i.e. Interior Point algorithm work to find bottom of the valley indicated by PSO.

Keywords: Parallel Robot, Flight Simulator, 6-RUS, Singularity, Optimization.

RESUMO

FURTADO, C. **Otimização da Orientação no Espaço de Trabalho de um Robo Paralelo 6-RUS**. 2015. 256 f. Dissertação (Mestrado em Engenharia Mecânica – Área: Projeto, Análise e Otimização de Sistemas Mecânicos) – Universidade do Estado de Santa Catarina, Programa de Pós-Graduação em Engenharia Mecânica, Joinville, 2015.

A Universidade do Estado de Santa Catarina construiu um simulador de voo de baseado robô paralelo axissimétrico com 6 DoF (graus de liberdade) que incorpora ambiente de realidade virtual de imersão. Este simulador de voo apresenta cadeia cinemática 6-RUS que é a segunda arquitetura mais comum para esta aplicação. Robôs paralelos são mecanismos de cadeia fechada que apresentam bom desempenho em termos de precisão, rigidez e capacidade para manipular grandes cargas. Este trabalho tem como objetivo otimizar um robô paralelo 6- RUS para determinar os melhores localizações das juntas ativas (posição e orientação) para uma tarefa de simulação de voo usando um algoritmo PSO (Particle Swarm Optimization) para escapar de mínimos locais e um algoritmo de Ponto Interior para acelerar a procurar pelo ótimo na região indicada por PSO ou seja o algoritmo de Ponto Interior trabalha para encontrar fundo do vale indicado pelo PSO.

Palavras-chave: Robô paralelo. Simulador de Voo. 6-RUS. Singularidade. Otimização.

LIST OF FIGURES

Figure 1 – ABB IRB 4400 serial robot.	30
Figure 2 – ABB IRB 340 FlexPicker parallel robot.	31
Figure 3 – 6-Rus parallel robot.	34
Figure 4 – IWF-robot.	35
Figure 5 – CEART-robot.	36
Figure 6 – Screw movement or twist.	37
Figure 7 – Twist components for a general screw kinematic pair. . .	38
Figure 8 – Wrench components.	43
Figure 9 – Parallel robot geometrical parameters.	45
Figure 10 – Vectorial chain to i -limb.	48
Figure 11 – Plane ω_i frontal view and required angles.	49
Figure 12 – Plane ω_i frontal view, geometrical details.	49
Figure 13 – Plane ω_i frontal view.	52
Figure 14 – Position of points B , and C	58
Figure 15 – (a) The wrenches acting upon the end-effector. (b) and (c) Two direct singularities.	63
Figure 16 – Grassmann varieties of dimension 1,2,3,4,5,6.	66
Figure 17 – Distance between two segments.	79
Figure 18 – a) Grassmann variety V5a on the Hexa; b) Power based index; c) Grassman variety V5b; d) Grassman V5b based index.	82
Figure 19 – Flowchart for particle swarm optimization algorithm. . .	88
Figure 20 – Internal Point Algorithm.	91
Figure 21 – 6-Rus Kinematic Chain.	98

Figure 22 – Base layout, where design parameters are $\chi_{a1}, \chi_{a2}, \beta_1, \beta_2,$ $\beta_3, \beta_4, \beta_5$ and β_6	99
Figure 23 – Active joint local system.	100
Figure 24 – Active Joint Angular Position θ	100
Figure 25 – Platform angular layout, where the design parameters are $\chi_{j1}, \chi_{j2}, r_p$ and e	101
Figure 26 – IWF-robot active joint location.	102
Figure 27 – Pitch-Roll-Yaw of an Air Plane	104
Figure 28 – Axis Orientation Rotation with α from 0° to 360°	105
Figure 29 – IWF-robot Polar Graphic.	106
Figure 30 – CEART-robot Polar Graphic.	107
Figure 31 – IWF-robot direct singularity index, see red arrow.	107
Figure 32 – IWF-robot inverse singularity constraint.	108
Figure 33 – CEART-robot direct singularity index.	109
Figure 34 – CEART-robot inverse singularity constraint, see red arrow. . . .	109
Figure 35 – Flowchart of MATLAB program optimization program.	110
Figure 36 – Optimized 6-RUS Polar Graphic.	113
Figure 37 – Direct singularity index for Y rotation, see red arrow.	114
Figure 38 – Passive links minimum distance.	115
Figure 39 – Cranks minimum distance, see red arrow.	115
Figure 40 – Inverse singularity constraint for Y rotation, see red arrow. . . .	116
Figure 41 – Crank 1 movement for Y rotation, see red arrow.	116
Figure 42 – Crank 6 movement for Y rotation, see red arrow.	117
Figure 43 – IWF-robot layout in ADAMS.	118
Figure 44 – IWF-robot torque application in initial position.	118

Figure 45 – IWF-robot torque application near direct singularity. . . .	119
Figure 46 – CEART-robot layout in ADAMS.	119
Figure 47 – CEART-robot rotation in Y axis, inverse singularity near 25°.	120
Figure 48 – Optimized 6-RUS layout in ADAMS.	121
Figure 49 – Optimized 6-RUS robot torque application near direct sin- gularity.	121
Figure 50 – Optimized 6-RUS movement for Y rotation.	121

LIST OF TABLES

Table 1 – 6-RUS Robot geometrical parameters.	44
Table 2 – PSO coefficients values.	89
Table 3 – IWF-robot angles parameters.	101
Table 4 – IWF-robot Geometrical Parameters.	102
Table 5 – CEART-robot angles parameters.	103
Table 6 – CEART-robot Geometrical Parameters.	103
Table 7 – Active joints orientation angle for 6-RUS optimized robot.	113
Table 8 – 6-RUS optimized parameters.	114

LIST OF ABBREVIATIONS AND ACRONYMS

HDBjoint	Half of Distance Between a Pair of Joints
HDBact	Half of Distance Between a pair of ACTuator
APact	Angles for position a Pair of ACTuator
APjoint	Angles for position a Pair of JOINTs
Aact	Angles for ACTuator's orientation
MDBL	Minimal Distance Between any pair of Limbs
MDBC	Minimal Distance Between any pair of Cranks

LIST OF SYMBOLS

r_b	Radius of the Base
r_p	Radius of the End-Effector
r_i	Length of the Crank
R_i	Length of the Passive Link
e	Half of Distance Between a Pair of Joints
d	Half of Distance Between a Pair of Actuators
χ_a	Angles for Position of Actuators
χ_j	Angles for Position of a Pair of Joints
β	Angles for Actuator's Orientation
θ	Active Joint Angular Position

CONTENTS

1	INTRODUCTION	29
2	PARALLEL ROBOTS	33
2.1	Robot Architectures	34
2.2	Screw Theory Representation	36
2.2.1	Wrench: action screw	40
2.2.2	Reciprocity and rate of work	42
2.3	Inverse Kinematic Problem	44
3	ROBOT SINGULARITIES	57
3.1	Direct Singularity	62
3.1.1	Singularity detection method	63
3.1.2	Grassmann Geometry	64
3.2	Singularity Closeness Measures	67
3.2.1	Linear Algebra Based Measures	68
3.2.2	Screw Theory Based Measures	69
4	SINGULARITY POWER INSPIRED MEASURE . . .	71
4.1	Minimization Problem	71
4.1.1	Twist Normalization: Invariant Norm	73
4.1.2	Objective function: power inspired measure	74
4.1.3	Constraints	78
4.1.3.1	Inverse Singularity	78
4.1.3.2	Cranks and limbs Collision	79
4.2	Corresponding eigenvalue problem	80

5	OPTIMIZATION	85
5.1	Particle Swarm Optimization	85
5.1.1	Particle Swarm Optimization with constriction factor	86
5.1.2	Configuration for Particle Swarm Optimization	88
5.2	FMINCON	89
5.2.1	Interior Point Algorithm	90
5.2.2	Sensitivity Analysis	92
5.3	Hybrid Optimization	94
6	ORIENTATION WORKSPACE OPTIMIZATION FOR A 6-RUS PARALLEL ROBOT	97
6.1	Kinematical Performance Index for Two 6 - RUS Parallel Robots	97
6.2	Geometrical Parameters of the IWF-robot and CEART- robot	101
6.3	IWF-robot and CEART-robot Performance for Flight simulator Task	103
6.3.1	Proposed task	103
6.3.2	IWF-robot and CEART-robot PERFORMANCE . . .	105
6.4	6-RUS Robot Optimization for Flight Simulator Task	108
6.5	Optimization Result and Kinematic Analysis . . .	112
6.5.1	Software Tool: Matlab	112
6.5.2	Software Tool: ADAMS	116
6.5.2.1	CAE Analyze: IWF-robot	117
6.5.2.2	CAE Analyze: CEART-robot	119
6.5.2.3	CAE Analyze: Optimized 6-RUS Robot	120

7	CONCLUSIONS	123
	REFERENCE	125
8	APPENDIX A: MATLAB ALGORITHM	131
8.1	Main Program	131
8.2	Program 1	135
8.3	Kinematic and Screw Based Program	136
8.4	Program 2	154
8.5	Function Objective Gradient	155
8.6	Constraints Program	159
8.7	Constraints Finite Differences Program	162
8.8	Constraints Gradient Program	167
9	APPENDIX B: ADAMS SCRIPT	169
9.1	ADAMS Parametric Model Script	169
9.2	Positions Points File	261

1 INTRODUCTION

This work aims to determine the optimal value for some geometrical parameters like active joints locations (position and orientation) for a 6-RUS parallel robot in order to maximize the orientation workspace i.e. end-effector, orientation.

Comparing parallel and serial robot configurations, it is observed that parallel robots are advantageous in terms of accuracy, stiffness and ability to manipulate large loads. Therefore, parallel robots are more suitable for tasks that require such properties. However the small workspace, compared to serial robots, is a drawback that researchers try to minimize, optimizing the parallel robot parameters (MERLET, 2012).

In machining applications, a parallel robot requires the capability to withstand large tool forces, which means that the robot structure must transmit forces with high accuracy and high stiffness. Different applications demand new requirement combinations which may be carry on optimizing the robot structure, for a given task. To keep a low cost level, it is infeasible to design customized robots for each application, therefore modularization is needed. Modularization and optimization strategies for robots based on parallel kinematics differ from the corresponding strategies used for robots based on serial kinematics. Therefore, optimization and modularization of parallel robots are interesting research tasks (BROGARDH, 2002).

As the performance of parallel robots is sensitive to their dimensions and geometry, a given design may be optimized varying these parameters. Among all kinematic measures, the workspace is one of the most important in-

Figure 1 – ABB IRB 4400 serial robot.



Source: (SICILIANO et al., 2010)

dices in the design of a parallel robot, (MERLET, 2012; KELAIAIA; ZAATRI et al., 2012; MERLET, 2006).

Stiffness optimization for a spatial 5-DOF parallel robot was developed using a genetic algorithm to escape from local minima (ZHANG, 2010). Barbosa, Pires and Lopes optimize the kinematic design a 6-dof parallel robot for maximum dexterity using a Genetic algorithm (BARBOSA; PIRES; LOPES, 2005). Stan, Maties and Balan applied a Genetic algorithm to multiple criteria optimization problems for 2-DOF micro parallel robot (STAN; MATIES; BALAN, 2007).

The parallel robot studied in this work is a flight simulator built

Figure 2 – ABB IRB 340 FlexPicker parallel robot.



Source: (SICILIANO et al., 2010)

at UDESC - Ceart (Centro de Artes) in Florianópolis, from now on named CEART-robot. This robot presents the second most common architecture for parallel robots used in flight simulators: the 6 - RUS. This work intends to establish a method for optimizing some robot geometrical parameters aiming at maximizing the orientation workspace i.e. the maximum plane orientation or twist in the flight simulator case. Aiming at optimizing this system a work-based index is used. This index measures the robot closeness to singularities, i.e. robot configurations where its degrees of freedom are modified. It is evaluated for different flight simulation configuration i.e. active joint location on the base, in order to optimized the maximum orientation into the workspace

(FURTADO; CAMPOS; REIS, 2014).

Proposed workspace (objective function) optimization is based on a kinematical index which is function of active joint location (optimization parameters). The methods used for this optimization are a Particle Swarm Optimization (PSO) algorithm and Interior Point algorithm, which is called by MATLAB optimization function FMINCON. Initially, the PSO algorithm optimizes, based on its metaheuristic behaviour, the problem to escape from local minima and provides a set of optimized parameters. The Interior Point algorithm, receives these optimized parameters and is employed to accelerate the search for bottom of the valley indicated by the PSO as the best particles locations.

Therefore this work provides a method for maximizing the workspace of 6 - RUS parallel robot by optimizing the robot active joints location, evaluating the singularity proximity and employing a hybrid approach, i.e. combining metaheuristic (PSO) and a derivative based algorithm (Interior Point).

The first step to develop this work is to determine the inverse kinematics equations, i.e. the relation between end-effector location and active joint rotation. Differential kinematics, i.e. the relation between end-effector velocity and active joint speed, and a singularity closeness index, used as constraint, are determined applying the screw theory. Using MATLAB, this work develops algorithms to perform the optimization and analyse the data. Optimized CAE (Computed Aided Engineering) model (using MSC-ADAMS) is developed to analyse the robot performance and evaluate the results. MATLAB and ADAMS are used to analyze the optimized 6-RUS robot configuration.

2 PARALLEL ROBOTS

Parallel robots, also named parallel manipulators, typically consist of a platform connected to a fixed base by several limbs (MERLET, 2001) (see Figures 3, 4 and 5).

A n -DOF (n -degree-of-freedom) fully-parallel mechanism is composed of n independent limbs connecting the end-effector to the fixed base. Each of these limbs is a serial kinematic chain that hosts one or more active joints which actuates, directly or indirectly, the end-effector (BONEV, 2003). Due to distribution of external load, parallel robots present good performances in terms of accuracy, rigidity and ability to manipulate large loads (MERLET, 2001).

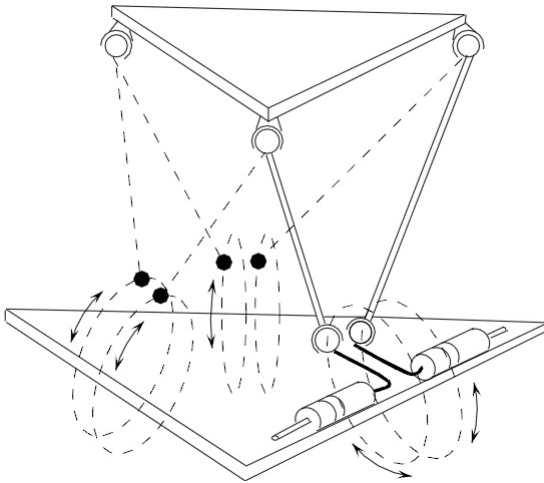
The 6-DOF parallel robot most studied architecture is the 6-UPS. This architecture is known as Stewart-Gough platform (FICHTER, 1986). The Stewart-Gough platform presents a stiff architecture because the load distribution is only axial and allows the use of powerful hydraulic actuators. Motion simulators, generally, manipulate excessive loads of up to tens of tons (BONEV, 2003).

The second most common architecture is the 6-RUS kinematic chain, this chain architecture was proposed by Hunt early in 1983 (MERLET, 2001). In this architecture the actuated joint is rotational, which leads to the interchange possibility of universal and spherical joint without any change in mechanism characteristics (BONEV, 2003). This work focuses on a method to optimize the workspace of the robot 6-RUS using an index of singularity closeness based on screw theory, (BALL, 1900; HUNT, 1983; CAMPOS, 2001).

2.1 Robot Architectures

As early as 1983 Hunt suggested a robotic architecture using this type of chain (HUNT, 1983) (see Figure 3). This parallel robot is composed of two platforms, one of them fixed to the ground, (base). On the base there are six rotating actuators R located on the edges of the triangle. These actuators are the input elements on which the active joints are located. Each actuator is linked to the end-effector through a rod. In each rod, one tip is linked to the crank of an actuator through an universal joint U and the other tip is connected to the end-effector by means of a spherical joint S , as depicted in Figure 3. A couple of rods converge on each spherical joint of the end-effector (ZABALZA et al., 2003).

Figure 3 – 6-Rus parallel robot.



Source: (MERLET, 2000).

The HEXA parallel robot, first presented in (PIERROT, 1990) is a

six degree of freedom DOF manipulator composed of six equally designed kinematic chains, which connect a base-platform to the end-effector platform. Each chain can be described as a serial RUS chain, where R stands for revolute joint, U for universal or cardan joint and S for spherical joint respectively (see Figure 21).

The revolute joints of the HEXA's structure are the active joints, which means that they are driven by an actuator while spherical joints in the structure remain passive. A prototype of the HEXA parallel robot has been designed and built at the Institute of Machine Tools and Production Technology (IWF) in Braunschweig, Germany, from now on named IWF-robot (see Figure 4), (LAST et al., 2005).

Figure 4 – IWF-robot.



Source: (LAST et al., 2005).

The flight simulator built at UDESC - Ceart (Centro de Artes) in Florianópolis also has a 6-RUS parallel robots configuration (see Figure 5).

Both configurations belong to the same family, so they have the same kinematic structure (FURTADO; CAMPOS; REIS, 2014).

Figure 5 – CEART-robot.



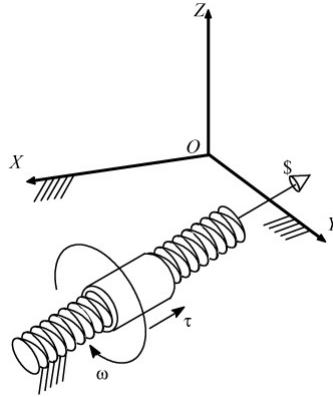
Source: Own Author.

2.2 Screw Theory Representation

The Mozzi theorem, states that the velocities of the points of a rigid body with respect to an inertial reference frame $O(X, Y, Z)$ may be represented by a differential rotation $\vec{\omega}$ about a certain fixed axis and a simultaneous differential translation $\vec{\tau}$ along the same axis. The complete movement of the rigid body, combining rotation and translation, is called screw movement or twist $\$$. Figure 6 shows the body "twisting" around an axis instantaneously fixed with respect of the inertial reference frame. This axis is called the screw axis and the rate of the translational velocity and the angular velocity is called the pitch of the screw $h = \|\vec{\tau}\|/\|\vec{\omega}\|$.

The twist represents the differential movement of the body with respect to the inertial frame and may be expressed by a pair of vectors, *i.e.* $\$ = [\vec{\omega} \ \vec{V}_p]^T$. The vector $\vec{\omega} = [\mathcal{L} \ \mathcal{M} \ \mathcal{N}]^T$ represents the angular velocity of

Figure 6 – Screw movement or twist.

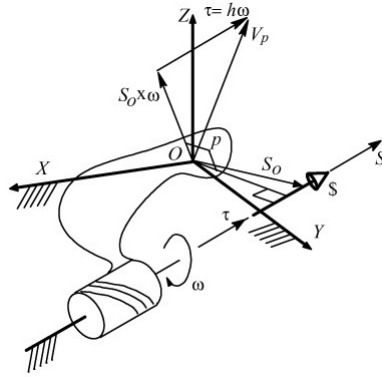


Source: (CAMPOS et al., 2011).

the body with respect to the inertial frame. The vector $\vec{V}_p = [\mathcal{P}^* \ \mathcal{Q}^* \ \mathcal{R}^*]^T$ represents the linear velocity of a point P attached to the body which is instantaneously coincident with origin O of the reference frame. If there are no points of the body coinciding with the frame origin O , as in Figure 6, a fictitious extension may be added to the body such that a point in this extension, named point P , coincides with the origin O (see Figure 7). The vector \vec{V}_p consists of two components: a) a velocity component parallel to the screw axis represented by $\vec{\tau} = h \vec{\omega}$; and b) a velocity component normal to the screw axis represented by $\vec{S}_o \times \vec{\omega}$, where \vec{S}_o is the position vector of any point at the screw axis.

A twist may be decomposed into its amplitude and its corresponding normalized screw. The twist amplitude Ψ is either the magnitude of the angular velocity of the body, $\|\vec{\omega}\|$, if the kinematic pair is rotative or helical, or the magnitude of the linear velocity, $\|\vec{V}_p\|$, if the kinematic pair is prismatic.

Figure 7 – Twist components for a general screw kinematic pair.



Source: (CAMPOS et al., 2011).

Consider a twist given by $\$ = [\vec{\omega} \ \vec{V}_p]^T = [\mathcal{L} \ \mathcal{M} \ \mathcal{N} \ \mathcal{P}^* \ \mathcal{Q}^* \ \mathcal{R}^*]^T$ then the correspondent normalized screw is $\hat{\$} = [L \ M \ N \ P^* \ Q^* \ R^*]^T$. This normalized screw is a twist in which the magnitude Ψ is factored out, i.e.

$$\$ = \hat{\$} \Psi \quad (2.1)$$

The normalized screw coordinates (HUNT, 2003) may be defined as a pair of vectors, named. $[L \ M \ N]^T$, dimensionless \vec{l} and $[P^* \ Q^* \ R^*]$, units

of length \vec{L} , given by,

$$\hat{\$} = \begin{bmatrix} L \\ M \\ N \\ P^* \\ Q^* \\ R^* \end{bmatrix} = \begin{bmatrix} \vec{S} \\ \vec{S}_o \times \vec{S} + h\vec{S} \end{bmatrix} \quad (2.2)$$

where \vec{S} is the normalized vector parallel to the screw axis. Notice that the vector $(\vec{S}_o \times \vec{S})$ determines the moment of the screw axis around the origin of the reference frame.

The movement between two adjacent links, belonging to a n -link kinematic chain, may be also represented by a twist. In this case, the twist represents the movement of link i with respect to link $(i - 1)$.

In robotics, generally, the differential kinematics between a pair of bodies is determined by either a rotative or a prismatic kinematic pair. For a rotative pair the pitch of the twist is null ($h = 0$). In this case the normalized screw or a rotative pair is expressed by

$$\hat{\$} = \begin{bmatrix} \vec{S} \\ \vec{S}_o \times \vec{S} \end{bmatrix} \quad (2.3)$$

For a prismatic pair the pitch of the twist is infinite ($h = \infty$) and the

normalized screw reduces to

$$\hat{\$} = \begin{bmatrix} 0 \\ \vec{S} \end{bmatrix} \quad (2.4)$$

2.2.1 Wrench: action screw

This section presents how an action (forces and moments) upon a body may be represented by a screw and a magnitude.

The *screw is a geometric element* composed by a directed line (*axis*) and by a scalar parameter h (length dimension) called *pitch*. If the directed line is represented by a normalized vector, the screw is called a *normalized screw* $\hat{\$}$.

The general action, *i.e.* a force and a couple, upon a rigid body in relation to a coordinate system is named a *wrench* $\$$, (HUNT, 2003). Any forces and moments system acting upon a rigid body (statics) may be reduced to a resultant force \vec{f} and a resultant couple \vec{C}_o , in relation to at choice system origin O , *i.e.* a wrench. In general, the resultant force vector and the resultant binary are not collinear. However, the system of forces and moments always can be reduced to a resultant force \vec{f} acting in the axis direction and a couple \vec{C}_{\parallel} , acting around the same axis (POINSOT, 1806).

A wrench may be represented by a scalar Ψ_r , representing the action magnitude, and by a normalized screw $\hat{\$}_r$, defined by the normalized vector in

the axis direction and by the pitch h_r , defined by

$$h_r = \frac{\|\vec{C}_{\parallel}\|}{\|\vec{f}\|} \quad (2.5)$$

For example, consider a static nut supporting a couple around its axis, corresponding screw axis, and additionally supporting the force, induced by the binary, in the axis direction. The action upon the nut may be declined by the scalar correspondent to the force magnitude (Ψ_r) and by the normalized screw composed by the normalized vector, in the screw axis direction, and by the pitch h_r , given by the rate between the binary and the force acting upon the nut.

The action upon a rigid body in relation to a coordinate system may be represented by a wrench composed by two vectors, *i.e.* $\$ = [\vec{f} \ \vec{C}_o]^T$, or in screw coordinate $[\mathcal{L}_r \ \mathcal{M}_r \ \mathcal{N}_r \ \mathcal{P}_r^* \ \mathcal{Q}_r^* \ \mathcal{R}_r^*]^T$, (HUNT, 2003). The vector $\vec{f} = [f_x \ f_y \ f_z]^T = [\mathcal{L}_r \ \mathcal{M}_r \ \mathcal{N}_r]^T$ represents the resultant force upon the body. The vector $\vec{C}_o = [C_{ox} \ C_{oy} \ C_{oz}]^T = [\mathcal{P}_r^* \ \mathcal{Q}_r^* \ \mathcal{R}_r^*]^T$ represents the resultant moment upon the body in relation to the coordinate system origin O . The vector \vec{C}_o is formed by two moment components: a) the moment component parallel to the screw axis represented by $\vec{C}_{\parallel} = h_r \vec{f}$; and b) the moment component normal to the screw axis represented by $\vec{C}_h = \vec{S}_o \times \vec{f}$ where \vec{S}_o , is the position vector of some point in on the screw axis, (see Figure 8). A wrench may be represented by its magnitude Ψ_r and by a normalized screw $\hat{\$}_r$ through

$$\$_r = \hat{\$_r} \Psi_r \quad (2.6)$$

The wrench magnitude Ψ , is the force magnitude $\|\vec{f}\|$, acting upon the body, if the action is a pure force, or it is the moment magnitude $\|\vec{C}_o\|$, if the action is a pure moment. If the action is a force and moment combination the wrench magnitude is $\|\vec{f}\|$. Considering a wrench $\$ _r = [\vec{f} \ \vec{C}_o]^T = [\mathcal{L}_r \ \mathcal{M}_r \ \mathcal{N}_r \ \mathcal{P}_r^* \ \mathcal{Q}_r^* \ \mathcal{R}_r^*]^T$, its corresponding normalized screw $\hat{\$}_r$ is defined by two vectors, $[L_r \ M_r \ N_r]^T$, dimensionless \vec{l} , and $[P_r^* \ Q_r^* \ R_r^*]^T$, units of length \vec{L} , so:

$$\hat{\$}_r = \begin{bmatrix} \mathcal{L}_r / \Psi_r \\ \mathcal{M}_r / \Psi_r \\ \mathcal{N}_r / \Psi_r \\ \mathcal{P}_r^* / \Psi_r \\ \mathcal{Q}_r^* / \Psi_r \\ \mathcal{R}_r^* / \Psi_r \end{bmatrix} = \begin{bmatrix} L_r \\ M_r \\ N_r \\ P_r^* \\ Q_r^* \\ R_r^* \end{bmatrix} = \begin{bmatrix} \vec{S}_r \\ \vec{S}_{O_r} \times \vec{S}_r + h_r \vec{S}_r \end{bmatrix} \quad (2.7)$$

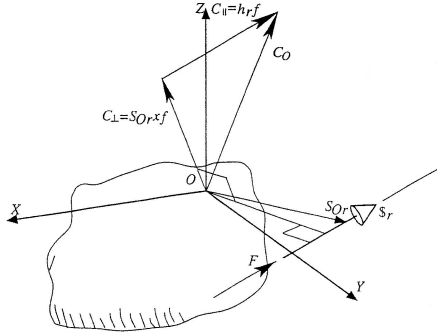
where \vec{S}_r is the normalized vector parallel to the screw axis. It is important to notice that the vector $\vec{S}_{O_r} \times \vec{S}_r$ the screw axis moment around the reference system origin.

2.2.2 Reciprocity and rate of work

If a non-null wrench ($\Psi_r \neq 0$) acts upon a rigid body in such way that it does not produce work while the body moves around a instantaneous twist ($\Psi \neq 0$), both screws (twist and wrench) are called *reciprocal screws* (BALL, 1900; HUNT, 2003).

Let a rigid body support a wrench $\$ _r = [\vec{f} \ \vec{C}_o] = \Psi_r \hat{\$}_r$ while it is

Figure 8 – Wrench components.



Source: (CAMPOS et al., 2011).

moving around a instantaneous twist $\$ = [\vec{\omega} \quad \vec{V}_p]^T = \Psi \hat{\$}$. Then, the rate of work or instantaneous power carried out is:

$$\delta W = \vec{C} \cdot \vec{\omega}_n + \vec{f} \cdot \vec{V}_p \quad (2.8)$$

For convenience, the transpose of a normalized screw is defined in Plucker axis coordinates (TSAI, 1999), by $\hat{\$}^T = [P^* \quad Q^* \quad R^* \quad L \quad M \quad N]$ and $\hat{\$}_r^T = [P_r^* \quad Q_r^* \quad R_r^* \quad L_r \quad M_r \quad N_r]$.

So, the rate of work is:

$$\delta W = \$_r^T \$ = \$^T \$_r \quad (2.9)$$

Additionally, the Equation 2.9 may be given by

$$\begin{aligned} \delta W &= (\hat{\$}_r^T \Psi) \$_r \\ \delta W &= (\hat{\$}_r^T \$_r) \Psi \Rightarrow \frac{\delta W}{\Psi} = \hat{\$}_r^T \$_r \end{aligned} \quad (2.10)$$

Table 1 – 6-RUS Robot geometrical parameters.

Symbol	Geometric Parameter
r_b	Radius of the Base
r_p	Radius of the End-Effector
r_i	Length of the Crank
R_i	Length of the Passive Link
e	Half of Distance Between a Pair of Joints (End-effector)
d	Half of Distance Between a Pair of Actuators (Base)
χ_a	Angles for Position of Actuators (Base)
χ_j	Angles for Position of a Pair of Joints (End-effector)
β	Angles for Actuator's Orientation (Base)

Source: Own author.

so, the reciprocity condition may be expressed as

$$\frac{\delta W}{\Psi} = 0 \quad (2.11)$$

due a non trivial case $\Psi \neq 0$, the reciprocity condition result in

$$\hat{\$}^T \$_r = 0 \quad (2.12)$$

2.3 Inverse Kinematic Problem

This section presents a method to solve the Inverse Kinematic Problem for a 6-RUS parallel robot using some non-physical parameters defined based on the vectorial equation of the chains.

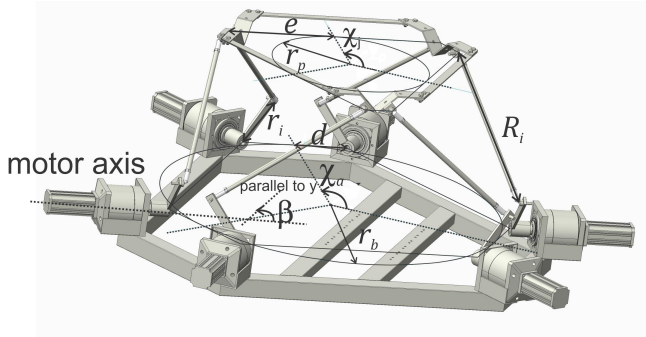
For the inverse kinematic problem, the vector describing the end-effector position in cartesian coordinates system and orientation by roll, pitch and yaw angles, respectively is given by $\vec{P} = [P_x \ P_y \ P_z \ \varphi \ \vartheta \ \psi]$.

In this method, few geometrical parameters are necessary to develop the inverse kinematic problem solution for a 6-RUS parallel robot. All can be

obtained from a prototype or a CAD model.

The required parameters are the fixed base radius (r_b), the end-effector radius (r_p), the crank dimension (r_i) which is located in an active joint, the passive link dimension (R_i), the half-distance between joints in the end-effector (e), the half-distance between an active joint pair (d), the position angle of the actuators pair (χ_a), the position angle of the passive spherical joint in end-effector (χ_j) and the angle between the crank rotational plane and an axis parallel to y (β). These parameters are detailed in Figure 9.

Figure 9 – Parallel robot geometrical parameters.



Source: Own author.

Initially it is needed to define the position of the actuators, which can be obtained by Equation 2.13, which must be solved for all limbs, resulting in a 6x3 vector:

$$\vec{A} = \vec{R} + \vec{m} \quad (2.13)$$

$$m = (-1)^{i-1} d \quad (2.14)$$

Being $i = 1, 2, 3, \dots, 6$ corresponding to each kinematic chain.

$$\vec{R} = r_b \cos \chi_a \hat{i} + r_b \sin \chi_a \hat{j} \quad (2.15)$$

$$\vec{m} = -m \sin \chi_a \hat{i} + m \cos \chi_a \hat{j} + 0\hat{k} \quad (2.16)$$

$$\vec{A} = (r_b \cos \chi_a - m \sin \chi_a) \hat{i} + (r_b \sin \chi_a + m \cos \chi_a) \hat{j} + 0\hat{k} \quad (2.17)$$

Similarly, all the platform joint positions, i.e. the spherical joints attached to the end-effector must be found in the local coordinate system attached to the base:

$$\vec{C} = \vec{r} + \vec{n} \quad (2.18)$$

$$n = (-1)^{i-1} e \quad (2.19)$$

$$\vec{r} = r_p \cos \chi_j \hat{i} + r_p \sin \chi_j \hat{j} + 0\hat{k} \quad (2.20)$$

$$\vec{n} = -n \sin \chi_j \hat{i} + n \cos \chi_j \hat{j} + 0\hat{k} \quad (2.21)$$

$${}^e\vec{PC} = r_p \cos \chi_j - m \sin \chi_j \hat{i} + r_p \sin \chi_j + m \cos \chi_j \hat{j} + 0\hat{k} \quad (2.22)$$

The rotational transformation using roll, pitch and yaw angles notation in (SCIAVICCO; SICILIANO, 1996) is used in order to find ${}^e\vec{PC}$ from Tool Center Point, attached to end-effector origin eO to spherical joint in general coordinate system attached to fixed base origin bO .

$$rot = \begin{bmatrix} c\varphi c\theta & c\varphi s\theta s\psi - s\varphi c\psi & c\varphi s\theta c\psi + s\varphi c\psi \\ s\varphi c\theta & s\varphi s\theta s\psi + s\varphi c\psi & s\varphi s\theta c\psi - c\varphi s\psi \\ -s\theta & c\theta s\psi & c\theta c\psi \end{bmatrix} \quad (2.23)$$

$${}^f\vec{PC} = rot {}^m\vec{PC} \quad (2.24)$$

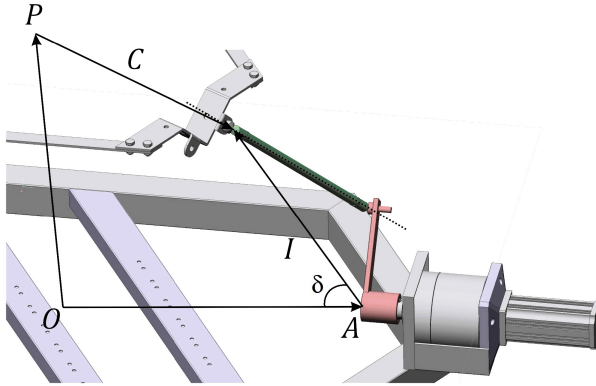
In order to find the vector from active joint to spherical joint of a limb ${}^b\vec{I}$, the vectorial Equation (2.25) must be solved.

$${}^b\vec{OA} = {}^b\vec{OP} + {}^b\vec{PC} + {}^b\vec{PC} - {}^b\vec{I} \quad (2.25)$$

$${}^b\vec{I} = {}^f\vec{OA} - {}^f\vec{OP} - {}^f\vec{PC} \quad (2.26)$$

The vector ${}^b\vec{I}$ may be decomposed into two components, \vec{I}_ω in the plane ω_i , and another one orthogonal to ω_i . \vec{I}_ω in ω_i may be further decomposed into two components, \vec{I}_z and \vec{T} , one parallel to the z axis and contained in plane ω_i and another one parallel to the xy plane, respectively.

$$\vec{I}_\omega = \vec{I}_z + \vec{T} \quad (2.27)$$

Figure 10 – Vectorial chain to i -limb.

Source: Own author.

The norm of vector \vec{T} may be written in the components of ${}^b\vec{T}$ terms, where I_x and I_y are ${}^b\vec{T}$ components in x and y axis, respectively.

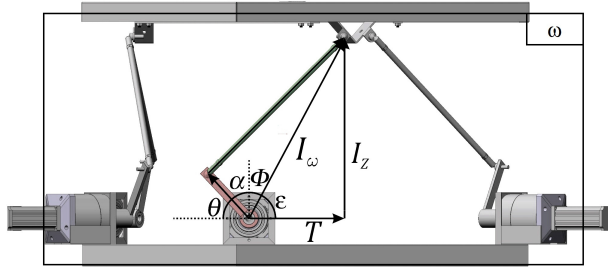
$$T = \|\vec{T}\| = I_x \cos \beta + I_y \sin \beta \quad (2.28)$$

With these definitions, it is possible to analyse the crank rotational plane ω_i and define the angles θ , α , ϕ and ε , which must be found to solve the inverse kinematic problem, being θ the crank angle.

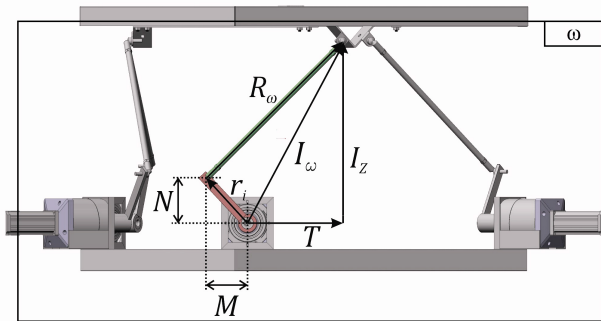
An analysis of Figure 11 leads to:

$$\pi = \varepsilon + \phi + \alpha + \theta \quad (2.29)$$

Using the inner product definition and calling M , horizontal component of \vec{r}_i in plane ω_i . Where $\frac{-\vec{T}}{\|\vec{T}\|} = -\hat{T}$.

Figure 11 – Plane ω_i frontal view and required angles.

Source: Own author.

Figure 12 – Plane ω_i frontal view, geometrical details.

Source: Own author.

$$\vec{r} \cdot -\vec{T} = \|\vec{r}\| \|\vec{T}\| \cos \theta \quad (2.30)$$

$$-\hat{T} = \hat{r} \quad (2.31)$$

$$\vec{T} \cdot \hat{r} = M \quad (2.32)$$

$$M = \|\vec{r}\| \cos \theta \quad (2.33)$$

Multiplying by $2T$ both sides.

$$2TM = 2T \|\vec{r}\| \cos \theta \quad (2.34)$$

A non-physical parameter u is defined which allows to re-write Equation 2.34 as:

$$u = 2Tr_i \quad (2.35)$$

$$u = \frac{2TM}{\cos \theta} \quad (2.36)$$

Using same method, another non-physical parameter v is defined, however the equation is multiplied by $2I_z$. Where N is the vertical component of \vec{r}_i and the formulation is similar to M .

$$v = -2I_z \|\vec{r}\| \quad (2.37)$$

$$\vec{r} \cdot \vec{I}_z = \|\vec{r}\| \|\vec{I}_z\| \cos \alpha \quad (2.38)$$

$$N = \|\vec{r}\| \cos \alpha \quad (2.39)$$

$$2I_z N = 2I_z \|\vec{r}\| \cos \alpha \quad (2.40)$$

$$v = \frac{2I_z N}{\cos \alpha} \quad (2.41)$$

In these terms is it possible to define the relation between $\frac{u}{v}$ and determine the angle ϕ .

$$\phi = \arctan \frac{T}{I_z} \quad (2.42)$$

$$\frac{u}{v} = \frac{2TM \cos \alpha}{2I_z N \cos \theta} \quad (2.43)$$

$$\frac{M}{N} = \frac{\cos \theta}{\cos \alpha} \quad (2.44)$$

$$\frac{u}{v} = -\frac{T}{I_z} \quad (2.45)$$

$$\phi = -\arctan \frac{u}{v} \quad (2.46)$$

It is not necessary to use this method to find ϕ once I_z and T are known. However, since the parameters u and v will be used in afterwards, it is convenient to show the formulation.

An analogous method is applied in order to find other relations to solve the inverse kinematic problem. Calling s the projection of I_ω over \vec{r}_i and multiplying by $2 r_i$ for convenience.

$$s = \left\| \vec{I}_\omega \right\| \cos(\phi + \alpha) \quad (2.47)$$

$$w = R_i^2 - \|\vec{I}\|^2 - r_i^2 \quad (2.51)$$

Also, manipulating the geometrical equations of the triangle rectangle showed in Figure 13, it turns out that.

$$s^2 = \left(\frac{-w}{2r_i} \right)^2 \quad (2.52)$$

$$u^2 + v^2 = (2Tr_i)^2 + (-2I_z r_i)^2 \quad (2.53)$$

$$u^2 + v^2 = (2r_i)^2 (I_\omega)^2 \quad (2.54)$$

From the triangle rectangle showed in Figure 13

$$z^2 + s^2 = \|\vec{I}_\omega\|^2 \quad (2.55)$$

$$(2r_i)^2 z^2 + (2r_i)^2 s^2 = (2r_i)^2 \|\vec{I}_\omega\|^2 \quad (2.56)$$

Then another non-physical parameter q , is defined and Equation 2.56 is re-written as.

$$q = 2r_i z \quad (2.57)$$

$$q^2 + (2r_i)^2 s^2 = (2r_i)^2 \|\vec{I}_\omega\|^2 \quad (2.58)$$

Substituting Equation 2.52 into Equation 2.58:

$$q^2 + w^2 = (2r_i)^2 \left\| \vec{I}_\omega \right\|^2 \quad (2.59)$$

$$q^2 = (2r_i)^2 \left(\left\| \vec{I}_\omega \right\|^2 - \left\| \vec{I}_\omega \right\|^2 \cos(\phi + \alpha) \right) \quad (2.60)$$

$$q^2 = (2r_i)^2 \left\| \vec{I}_\omega \right\|^2 \sin(\phi + \alpha) \quad (2.61)$$

$$\frac{q^2}{w^2} = \frac{(2\|\vec{r}\|)^2 \left\| \vec{I}_\omega \right\|^2 \cos^2(\phi + \alpha)}{(2\|\vec{r}\|)^2 \left\| \vec{I}_\omega \right\|^2 \sin^2(\phi + \alpha)} \quad (2.62)$$

$$\tan(\phi + \alpha) = \frac{q}{w} \quad (2.63)$$

$$\phi + \alpha = \arctan \frac{q}{w} \quad (2.64)$$

Returning to Equation 2.29.

$$\pi = \varepsilon + \phi + \alpha + \theta$$

$$\varepsilon = \frac{\pi}{2} - \phi \quad (2.65)$$

$$\pi = \left(\frac{\pi}{2} - \phi \right) + (\phi + \alpha) + \theta \quad (2.66)$$

$$\theta = \frac{\pi}{2} + \phi - (\phi + \alpha) \quad (2.67)$$

$$\theta = \frac{\pi}{2} - \arctan \frac{u}{v} - \arctan \frac{q}{w} \quad (2.68)$$

With these definitions, the inverse kinematic problem is solved. Once function \arctan has a dubious response, it is convenient to use \arctan with two arguments (*atan2* function).

An algorithm that compute the joint and actuators position and compute the parameter u , v , w and q as presented solve the inverse kinematic problem using really low computational coast.

For convenience Equations 2.35, 2.37, 2.51, 2.59 and 2.54 may be re-written to compute u , v , w and q based in previously values and apply in 2.68.

$$u = 2r_i (I_x \cos \beta + I_y \sin \beta) \quad (2.69)$$

$$v = -2r_i I_z \quad (2.70)$$

$$w = R_i^2 - r_i^2 - I_x^2 - I_y^2 - I_z^2 \quad (2.71)$$

$$u^2 + v^2 = q^2 + w^2 \quad (2.72)$$

$$q = \sqrt{u^2 + v^2 - w^2} \quad (2.73)$$

The same algorithm is useful to find the workspace to a given orientation using some increments to P_x, P_y and P_z and solving the Inverse Kinematic Problem repeatedly it is possible to find the workspace of the parallel robot to a given orientation.

3 ROBOT SINGULARITIES

The kinematics study of mechanical systems leads inevitably to the singular configurations problem. They correspond to configurations of the system that are usually undesirable since the degree of freedom is instantaneously changed. These special configurations are defined as the ones in which the Jacobian matrix, i.e., the matrix relating the input rates to the output rates, becomes rank deficient (GOSSELIN, 1988).

In this section it is present a differential kinematic relation for parallel manipulators and introduce their singularities. This differential kinematics is based on the parallel manipulator Jacobian matrix.

In spatial parallel manipulators, the relationship between actuator coordinate vector q and end-effector Cartesian coordinate vector P , may be stated as a function f

$$f(\theta, P) = 0 \quad (3.1)$$

where 0 is the 6-dimensional null vector, Therefore, the differential kinematic relation may be determined (TSAI, 1999)

$$\begin{aligned} J_q \dot{\theta} - J_x \$ &= 0 \\ J_q \dot{\theta} &= J_x \$ \\ \dot{\theta} &= J \$ \end{aligned} \quad (3.2)$$

where $\$$ is the end-effector velocity in ray order, $\dot{\theta} = [\Psi_1, \dots, \Psi_l]$ is the input twist magnitude vector and $J = J_q^{-1} J_x$ is the Jacobian matrix of the manipulator composed by direct J_x , and inverse J_q Jacobian matrices.

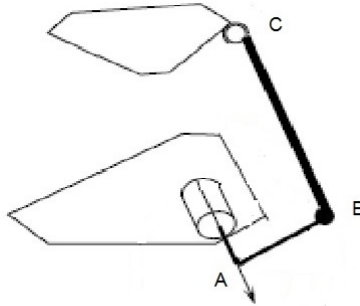
Additionally, we may write 3.2 as at differential kinematic relation-

ship between the end-effector velocity \dot{x} and the vector $v = [v_1, \dots, v_n]^T$

$$v = J_x \dot{x} \quad (3.3)$$

where v is the component of the absolute linear velocity of the end-effector connection point in the direction of the passive link, *i.e.* the distal link of each limb (serial chain between basis and end-effector) (DAVIDSON; HUNT, 2004a), *e.g.* in the 6-RUS parallel robot of Figure 14, the connection point is C_i and the passive link is $\overline{B_i C_i}$.

Figure 14 – Position of points B , and C .



Source: (MERLET, 2000).

Singular configurations occur if either J_x or J_q is singular. If J_q is singular, a *limb* or *inverse kinematics* singularity is encountered and end-effector is over constrained (TSAI, 1999), *i.e.* it instantaneously loses at least one degree of freedom.. Using equation 3.4, there is a inverse kinematic singularity if there exists a non-zero input velocity, $\Psi_a = \dot{\theta}$, which results in a

zero output, $\dot{\theta} = 0$. In this case:

$$J_q \dot{\theta} = 0 \quad (3.4)$$

In other words, this type of singularities consists of the set of *points where different branches of the inverse kinematic problem meet*, the inverse kinematic problem being understood here as the computation of values of the input variables from given values of output variables (GOSSELIN, 1988).

The inverse kinematic singularity is found, typically, at the boundary of the workspace or when the limbs folds upon itself. This type of singularity is caused due the serial nature of the limbs and is discussed extensively in literature (SCIAVICCO; SICILIANO, 1996; TSAI, 1999).

If J_x is singular, a platform or direct kinematic singularity is encountered and the end-effector can move instantaneously even if all actuators are locked. At these configurations, the manipulator gains one or more uncontrollable degrees of freedom at the end-effector. For instance, there exists a non-zero output velocity, $\dot{\theta}$, corresponding to a zero input velocity, $\dot{\theta}$, *i.e.*

$$0 = J_x \dot{\theta} \quad (3.5)$$

This corresponds to configurations in which the chain remains uncontrollable even when all the actuated joints are locked. As opposed to first one, this type of singularity lies within the workspace of the chain and corresponds to a *point or set of points where different branches of the direct kinematic problem meet*. The direct kinematic problem is the one in which it is desired to obtain the values of the output variables from given values of the input

variables. Since the nullspace of J is not empty, there exists a set of output rate vectors \dot{x} which will be mapped into the origin by J , i.e., which will correspond to a velocity of zero of the input joints. The input rates are therefore not independent (GOSSELIN, 1988). This type of singularity occurs within the workspace and is the main goal of this report.

A third type of singularity happens when both J_x and J_q are singular. This situation was first inadvertently classified as dependent on a certain special design of the manipulator, but was later proven incorrect (DANIALI; ZSOMBOR-MURRAY; ANGELES, 1995) by way of examples that the third type of singularity does not occur only in "special" mechanism. Rather, it can occur in "regular" mechanism.

In general inverse kinematic singularities are easier to detect and move outside of the desired workspace by changing limb lengths. However, since direct kinematic singularities happen within the workspace, they effectively partition the workspace into smaller usable portions.

There are three basic reasons why direct singularities become an issue in real life situations. They follow directly from the interpretations of singularities and are: reduced accuracy, large internal forces and loss of knowledge of solution tree (VOGLEWEDE, 2004).

- Degenerate accuracy. The Jacobian J ($J = -J_q^{-1}J_x$) (TSAI, 1999) analysis relates the input and output velocities. At direct singularity the end-effector can have an instantaneous velocity at the output for zero input velocity.

If the differential inputs and outputs are taken, the Jacobian relations

becomes

$$\begin{aligned}\dot{\theta} &= J\dot{X} \\ \frac{\Delta\theta}{\Delta t} &\approx J \frac{\Delta X}{\Delta t} \\ \Delta\theta &\approx J\Delta X\end{aligned}\tag{3.6}$$

Therefore, theoretically, when the manipulator is at a direct singular pose, the end-effector only has a very small instantaneous motion, *i.e.* no motion at all. However, in practice, all manipulators have some amount of clearances (and similarly some compliance) and allow some finite motion at the end-effector. This motion is denoted as the unconstrained end-effector motion. The problem is to find out how much unconstrained end-effector motion is allowed.

- Large internal forces. The direct kinematic singularities affect the internal forces, *i.e.* the static forces required by the actuators (input forces) approach to infinity at singularities.

- Solution tree. The solutions for the forward kinematics coincide at direct singularities. In practice the two different solutions cause control problems. If the manipulator moves close to a singularity and slips into the other solution due to overshoot, the manipulator mechanically is not where the control believes it is. The manipulator will be at a different end-effector position than predicted and could lead the manipulator into poses where it was not intended to operate.

3.1 Direct Singularity

Direct singularities may be introduced studying the mechanical equilibrium of a parallel robot. Consider the input articular torque/force vector $\tau = [\Psi_{r1}, \dots, \Psi_{r6}]$, *i.e.* the input wrench magnitude vector. If a wrench $\$r$ is applied on the end-effector, the mechanical system will be in equilibrium, if the resultant of the articular forces $(\$_{r1}, \dots, \$_{r6})$ which act on the platform is equal and opposite to the $\$r$, if not the end-effector will move to reach an equilibrium position. In an equilibrium position there is a relation between τ and $\$r$ (DAVIDSON; HUNT, 2004b)

$$J_x^T \tau = \$r \quad (3.7)$$

where the columns of J_x^T are the normalized screws (axial order) representing the wrenches acting on the end-effector(end-effector), *i.e.* the Plucker vector coordinates of the actuating force lines (MERLET, 2000; TSAI, 1999).

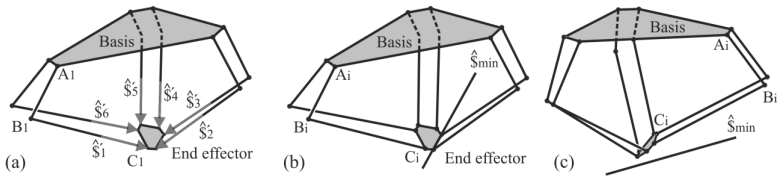
$$J_x^T = \begin{bmatrix} L_{r1} & \dots & L_{r6} \\ M_{r1} & \dots & M_{r6} \\ N_{r1} & \dots & N_{r6} \\ P_{r1}^* & \dots & P_{r6}^* \\ Q_{r1}^* & \dots & Q_{r6}^* \\ R_{r1}^* & \dots & R_{r6}^* \end{bmatrix} = \begin{bmatrix} \hat{\$}_{r1} & \dots & \hat{\$}_{r6} \end{bmatrix} \tau \quad (3.8)$$

The quantity $\$r$, is at wrench screw representing the action supported by the body, in ray coordinates. From this equation, it is seen that not only does the Jacobian relate the input and output velocities, but it relates the static

input to static forces and moments. Therefore, there is another interpretation of singularities.

Figure (15a) shows the wrenches acting upon the end-effector. Two examples of possible direct singularities are shown in Figure (15b) and (15c), these are two cases previewed by Grassmann analysis that are inside the robot workspace.

Figure 15 – (a) The wrenches acting upon the end-effector. (b) and (c) Two direct singularities.



Source: (CAMPOS; GUENTHER; MARTINS, 2005).

Direct kinematic singularities occur when there exists a force/moment, in the end-effector, in one or more directions that cannot be resisted by the inputs. When all the forces, acting up on the end-effector, intersect a one point the manipulator cannot resist a moment around that point. This point may be infinity and in this case the forces are parallel and they cannot resist a perpendicular force acting on the end-effector.

3.1.1 Singularity detection method

Singular configurations rest on the singularities of the Jacobian matrix. Singularities correspond to the roots of Jacobian determinant. However, the calculation of this determinant, even using symbolic algebra software,

e.g. Maxima (www.gnu.org/software/maxima/maxima.html), is a complicated labour for general robots (MERLET, 2000). After the calculating, of the determinant it is necessary to find its roots within the workspace which is a even more difficult task because the determinant in general is a non linear expression. This method is useful only for particular parallel robots (DANIALI; ZSOMBOR-MURRAY; ANGELES, 1995; TAHMASEBI, 1992).

Some researchers analyse intuitively some particular cases of singularity for the Jacobian matrix and have obtained certain cases of singularity (FICHTER, 1986), (HUNT, 1978), (LIU et al., 1993). Geometric method, based on Grassmann geometry, solve the problem for many different robots (MERLET, 2000). There are other methods which use a special variable to "measure" how "far" a pose is from a singular, for instance the *condition number* of the Jacobian matrix (XU; KOHLI; WENG, 1994). In this section, the geometric method and the closeness to singularity measures are presented.

3.1.2 Grassmann Geometry

Equation 3.8 is a linear system of equations in term of articular force τ . If the system is rigid, this means that for any action $\$_{\tau}$ (force moment) up on the end-effector, there exists one set of articular forces τ (or moments) such that the system is in an equilibrium state. This relation only is possible if matrix J_x^T is of full rank, i.e. the Plucker vectors (columns of J_x^T) are linearly independent. So, *a singular configuration of a parallel manipulator corresponds to a configuration where it is no rigid*.

In most cases the wrenches acting upon the parallel manipulator end-effector are pure forces (null pitch screw) and their screw components cor-

respond to a Plucker components of a line. *i.e.* a Plucker vector. In these cases, the singular configurations of the manipulator are associated with linearly dependent set of lines, also called line based singularities (HAO; MCCARTHY, 1998). Grassmann studied the varieties of lines. *i.e.* the sets of linear dependent lines to n given independent lines, and characterised them geometrically (MERLET, 1989).

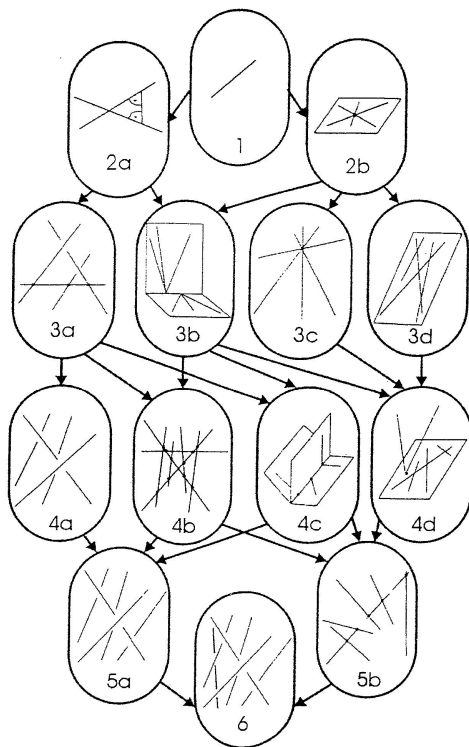
These varieties were analysed and classified in order to study the singularities of spatial frameworks (DANDURAND, 1984) and triangular simplified symmetric manipulator whose six limbs connect to the end-effector and to the basis only in three points, respectively (MERLET, 2000)(HAO; MCCARTHY, 1998).

The linear system of equations Equation (4.7) represents a rigid system if for any action $\$_r$ upon the end-effector, it is possible to find a set of input wrench magnitudes F_i . This condition is only feasible when matrix J_x^T is full rank, *i.e.* the Plucker vectors (columns of J_x^T) are linearly independent. So a singular configuration of a parallel manipulator corresponds to a configuration where it is not rigid.

The linear varieties are classified according to its rank (MERLET, 1989)(see Figure 16):

- The zero rank set is empty.
- The variety of rank one contains a line in the 3D space. The variety of rank two is composed by either a pair of skew lines in \mathbb{R}^3 , *i.e.* lines which

Figure 16 – Grassmann varieties of dimension 1,2,3,4,5,6.



Source: (MERLET, 2000).

have no intersection and are not parallel (also called agonic lines), or a flat pencil of lines, *i.e.* lines in a plane which intersect in one point.

- The variety of rank three is of four types: a regulus 3a, two flat pencil of lines in different planes and with different centres 3b, a bundle of lines (all lines through a point) 3c, and all lines in a plane 3d.

- The varieties of rank four, called linear congruences, are of four cases: elliptic, hyperbolic, parabolic and degenerated congruence, details in (HUNT, 1990; MERLET, 1989).

- Linear varieties of rank five, called linear complex, are of two types: nonsingular (or general): generated by 5 independent skew lines 5a, the complex is the set of lines tangent to coaxial helices; and singular (or special): all the lines meeting one given line 5b, this case is subdivided depending on if this line is at infinity 5b2 or not 5b1 (HAO; MCCARTHY, 1998).

Therefore, if the Plucker vectors, lines in the direction of the forces acting upon the end-effector, are in any of the above varieties (rank= 1, ..., 5), the parallel manipulator is in a direct singular configuration.

3.2 Singularity Closeness Measures

In this section some methods to measure the closeness of the singularity are presented based on the Voglewede thesis (VOGLEWEDE, 2004).

Initially, the criteria of measure must be *frame invariant, scale invariant and unit invariant*:

- **Frame invariant.** A quantity that does not change due to a change of the coordinate system's location or orientation is said to be frame invariant.

- **Unit invariant.** A quantity does not change due a change in the

units of the coordinate frame (*e.g.* meters or inches) is said to be unit invariant.

- **Scale invariant.** A quantity does not change due to a change of the size (or scaling) of the manipulator is said to be scale invariant.

3.2.1 Linear Algebra Based Measures

Mathematically, singularities are defined where the Jacobian relationship degenerates. Therefore, the most obvious method to evaluate singularities is to use techniques from linear algebra.

Condition number

The condition number was first suggested by Yoshikawa (YOSHIKAWA, 1990) as a local measure of how close one is to singularity. The condition number is defined as

$$k = \frac{\sigma_{min}}{\sigma_{max}} \quad (3.9)$$

where σ_{min} and σ_{max} are the minimum and the maximum singular values of the Jacobian matrix, J . This measure is not frame or unit invariant if the entries of the Jacobian matrix are not uniform (LIPKIN; DUFFY, 1988; DUFFY, 1990).

Jacobian Determinant

Another possible measure to know how close one is to a singularity is the determinant of the Jacobian matrix. It is proved that the determinant of the Jacobian matrix is frame and unit invariant (MURRAY; LI; SASTRY, 1994).

However, the geometric meaning of the determinant is not clear, *i.e.* we do not know what is a “good” value for determinant. Additionally, in general, finding a physical meaning for the determinant is very difficult (VOGLEWEDE, 2004).

3.2.2 Screw Theory Based Measures

Screws are briefly introduced in Chapter 2. This analysis decomposes all instantaneous rigid body motion into a rotation around an axis and a translation along the same axis, *i.e.* a twist. A dualism of this concept is that all forces acting on a rigid body can be decomposed as a force along an axis and a moment around that axis, *i.e.* a wrench.

Using the screw theory, specifically the rate of work presented in Section 2.2.2 a technique was developed to determine how close one is to a singularity (POTTMANN; PETERNELL; RAVANI, 1998). The rate of work or power product is defined as the normal dot product between a wrench $\$$, represented in axis coordinates, and a twist $\$$, represented in ray coordinates, or vice versa, see Equation 2.8 here repeated.

$$\delta W = \$^T_r \$ = \$^T \$_r = C \cdot \omega_n + f \cdot V_p \quad (3.10)$$

where δW is the instantaneous power between the force and the twist. Portman *et al.* determine the twist that minimize the square of the power (to keep two different power calculations from canceling out) with all the rows of the Jacobian (*i.e.* the wrenches). This based power calculation is used as a measure to see how close one is to a singularity. In other words, they find the twist that goes against the constraints the least, and then calculate the power via

the power product, Equation 3.10. This approach is used for a linear complex approximation technique to measure the closeness to singularities (WOLF; SHOHAM, 2003a) and it is presented in the next chapter.

4 SINGULARITY POWER INSPIRED MEASURE

This chapter determines closeness to singularities by formulating the question in terms of a constrained optimization problem. The constrained optimization problem results in a corresponding generalised eigenvalue problem. The resulting eigenvalue has physical meaning and is utilized as a measure of the performance near singularities, and thus is a measure of closeness to singularities.

This optimization approach was used to the singularity analysis. Pottman et al.(POTTMANN; PETERNELL; RAVANI, 1998) use the constrained optimization problem to determine the linear complex that is closest to a singularity. Wolf and Shoham (WOLF; SHOHAM, 2003a) use the methodology to describe the instantaneous behaviour near singularities. Voglewede (VOGLEWEDE, 2004) incorporate several other seemingly non-related measures into the constrained optimization framework, for instance the natural frequency measure.

4.1 Minimization Problem

Accordingly Voglewede (VOGLEWEDE, 2004), a measure of closeness to singularities $M(\mathbf{X})$, at a particular configuration, \mathbf{X} , *i.e.* position and orientation,

$$M : \{\text{configuration space}\} \rightarrow \mathbb{R} \quad (4.1)$$

$$M : \{\mathbf{X}\} \rightarrow \mathbb{R}$$

should have the following three properties:

- $M(\mathbf{X}) = 0$ if and only if \mathbf{X} is a singular configuration,
- If \mathbf{X} is non-singular, $M(\mathbf{X}) > 0$, and
- $M(\mathbf{X})$ has clear physical meaning.

If one approaches the problem of creating a measure from a physical standpoint, one first need to define the physical quantity, $M(\mathbf{X})$, one cares about. As shown above, a singular configuration has many different effects, including loss of constraint, increased end-effector error, loss of stiffness and degenerated actuator torque transmission. The effect that we are most concerned about is the loss of constraint so it forms the basis of the singularity measure.

In this case, if one cares primarily about the loss of constraint that occurs in at least one direction at a singular configuration, an appropriate measure that also applies to non-singular configurations is the amount of constraint provided by the mechanism in the least constrained direction. In this case, the physical quantity of interest at configuration \mathbf{X} for twist direction $\$$ is denoted by a value of the physical quantity $F(\mathbf{X}, \$)$, a meaningful measure

is obtained by minimizing F over all "unity" motions, $\$$, as follows:

$$M(\mathbf{X}) = \begin{cases} \min & F(\mathbf{X}) \\ \text{subject to} & \| \$ \|^2 = c \end{cases} \quad (4.2)$$

where $\| \dots \|$ represents a type of norm for twists (several norms may be found in (VOGLEWEDE, 2004) and c is a constant. The twist is constrained to have a certain norm because otherwise the trivial solution, $\$ = 0$, would always be the minimum.

So it is necessary to obtain an objective function F and an appropriate normalization of $\$$ for minimization the problem of Equation 4.2.

4.1.1 Twist Normalization: Invariant Norm

The general minimization problem presented in Equation 4.2 requires normalization a of a twist, which raises many issues on how this vector should be normalized, due the different dimension terms in the vector (twist) components. Specifically, we chose the invariant norm of the twist, other most used common norms are the Euclidean norm and the kinetic energy norm (VOGLEWEDE, 2004).

The *invariant norm* takes the magnitude of only the frame-invariant portion of the screw, *i.e.* the angular velocity component of the twist. so for a

given twist:

$$\$_ = \begin{bmatrix} \omega \\ V_p \end{bmatrix} \quad (4.3)$$

the invariants norm is

$$\| \$_ \| = \sqrt{\omega \cdot \omega} = \sqrt{\$_^T D \$_} \quad (4.4)$$

where

$$D = \begin{bmatrix} I_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} \end{bmatrix} \quad (4.5)$$

It is important to notice that if $\omega = 0$, the invariant norm becomes null, therefore for the case of pure translation the invariant norm must be supplemented, for instance a potential solution is (VOGLEWEDE, 2004):

$$\text{If } \omega = 0, \text{ then } \| \$_ \| = \sqrt{V_p \cdot V_p} \quad (4.6)$$

and in this case

$$D = \begin{bmatrix} 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & I_{3 \times 3} \end{bmatrix} \quad (4.7)$$

The invariant norm of Equation 4.4 deals well with finite pitch twists ($h \neq \infty$), i.e. twists with $\omega \neq 0$.

4.1.2 Objective function: power inspired measure

A objective function which results of the power inspired measure is used to state the minimizing problem (POTTMANN; PETERNELL; RAVANI,

1998; WOLF; SHOHAM, 2003b)

$$F(\$) = \sum_{i=1}^k (\hat{\$}_{ri}^T \$)^2 \quad (4.8)$$

here $\hat{\$}_{ri}$ are the columns, in axial order, of J_x^T , see Equation 3.8 that, particularly, can be interpreted as unitary magnitude wrenches $\hat{\$}_{ri} \cong \$_{ri}$, and k is the number of rows. The motivation for this measure, using parallel manipulators, comes from the dual meaning of the Jacobian matrix. Namely, columns of J_x^T may be interpreted as the (normalized) wrenches applied by the limbs (specifically by passive links) for unit actuator torques and each term $(\hat{\$}_{ri}^T \$)^2$ of F may be interpreted as the square of the power e.g. $[N^2 \cdot m^2/seg^2]$ of the i th limb on the end-effector twist, $\$$. Strictly, given that $\hat{\$}_{ri}$ is a normalized screw and is not a unitary magnitude wrench, the magnitude of F is L/T^2 e.g. $[m/seg]$ because, see Equation 3.3.

$$\begin{aligned} \hat{\$}_{ri}^T \cdot \$ &= [P_{ri}^* Q_{ri}^* R_{ri}^* L_{ri} M_{ri} N_{ri}] \cdot [\mathcal{L}_r, \mathcal{M}_r, \mathcal{N}_r, (\mathcal{P}^*, \mathcal{Q}^*, \mathcal{R}^*)]^T \\ &= F \\ &= (J_x \$)^T (J_x \$) \\ &= v^T v \\ &= \|v\|^2 \end{aligned} \quad (4.9)$$

In a singular configuration there exist a twist $\$$ for which none of the limb actions can do any work and thus the minimum of F goes to zero. Away from singular configuration, the minimization identifies the *least constrained twist* - with the restriction that its angular part be of norm one - and uses the power done by the constraining limb forces on that twist as a measure.

Rearranging Equation 4.8 in the form of Equation 4.2, this measure becomes:

$$M(\mathbf{X}) = \begin{cases} \min & F(\$) = \$^T J_x^T J_x \$ = \$^T M \$ \\ \text{subject to} & g(\$) = \$^T D \$ - c = 0 \end{cases} \quad (4.10)$$

where M (also called Graminiam matrix) and D are $n \times n$ symmetric positive semi-definite matrices, g is a given constraint, and c is some positive constant (VOGLEWEDE, 2004), therefore F only takes on non-negative values, $F(\$) \geq 0$.

We want to minimize the function F subject to given constraint. In other words, into the set of values of $\$$ that satisfy the constraint, we want to find the ones that give the minimum value F .

All possible constraints for a two variable function, can be rearranged to read $g(x, y) = 0$. For instance, the constraint $x^2y = 3$ may be written as $x^2y - 3 = 0$ and so forth. The equation $g(x, y) = 0$ give us a line or curve in the xy plane. We want to go along that line and find the point on with the largest value of F (in this example $F(x, y)$). When we find the largest value, we know that $F(x, y)$ will be stationary as we move along the line. That is what a minimum (or maximum) means. This in turn means that the gradient ∇F is perpendicular to the line of the constraint at this point, because the gradient is always perpendicular to the contour. But the constraint is also constant along the line, it is zero by construction. So it also has a gradient ∇g that is perpendicular to the line. Thus the two gradients are not equal but they

do point in the same direction. In other words, they are proportional to each other $\nabla F = \lambda \nabla g$, where λ is some number that we do not at present know the value of. This number λ is called a *Lagrange multiplier*. Rearranging we see that $\nabla F - \lambda \nabla g = 0$, or if given that:

$$\bar{F}(x, y) = F(x, y) - \lambda g(x, y) \text{ then } \nabla \bar{F} = 0 \quad (4.11)$$

If $\nabla \bar{F} = 0$, the all partial derivatives of \bar{F} are zero. This give us a method to finding our minimum. We simply construct the function $\bar{F}(x, y)$ as above, but multiplying g by a constant λ , then we set all partial derivatives of $\bar{F}(x, y)$ to zero and solve the set of simultaneous equations we get. This give us a solution for the minimum of \bar{F} .

The only problem is that the solution still contains the constant λ , whose value we do not know. But we can solve that problem easily enough. We still have one more equation - the constraint. If we substitute our solution for x and y back into the constraint, we get another equation that we can solve for λ .

It is possible to show that the method generalizes to more than two variables and more than one constraint. If we have a set $\{x_i\}$, $i = 1, \dots, n$ of variables and set $g_j(\{x_i\})$, $j = 1, \dots, m$ of constraints, then one constructs the function

$$\bar{F}(x_i) = F(x_i) - \sum_{j=1}^m \lambda_j g_j(x_i) \quad (4.12)$$

where $\{\lambda_j\}$ is a set of m unknown Lagrange multipliers. Then we solve the n

simultaneous equations

$$\frac{\partial \bar{F}}{\partial x_i} = 0 \quad (4.13)$$

to get x_i in terms of the λ_j , and we substitute the answer back into the m constraints to get m more equations that we solve for λ_j .

4.1.3 Constraints

Constraints are imposed conditions that variables must to satisfy and solution is a set of variable value that satisfies all constraints i.e., a point in the feasible region.

4.1.3.1 Inverse Singularity

Inverse singularity is caused due the serial nature of the limbs and will occur in 6-RUS parallel robots if one or more of the six limbs are fully extended or fully contracted. This means that the parallel robot loses one or more degrees of freedom, banning some moves.

Inverse singularity proximity may be analyzed by alignment between the crank and the passive link of each limb. This analysis may be done from the kinematic equations. When inverse singularity occurs the root of the Equation 4.14 is negative (q is an imaginary number), which means this limb can not be more extended or contracted because of crank and passive link lengths.

$$q = \sqrt{u^2 + v^2 - w^2} = \sqrt{-\mu} \quad (4.14)$$

So the singularity proximity may be evaluated by checking how close this root is next to become negative. Aiming avoid this the sum of geometric

terms u^2 and v^2 must be greater than w^2 . Thus a proximity inverse singularity index Inv_I of each limb may be defined as:

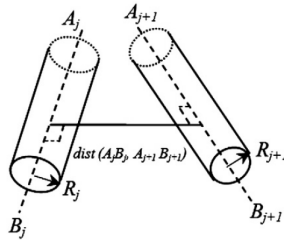
$$Inv_I = \frac{u^2 + v^2}{w^2}, Inv_I > 1 \quad (4.15)$$

4.1.3.2 Cranks and limbs Collision

Collision avoidance guarantees the orientation aptitudes of the robot. It may be denote by R_j, R_{j+1} the radius of two cylindrical segments $A_j B_j$ and $A_{j+1} B_{j+1}$, Figure 17, we can ensure that there is no mechanical interference if the distance between any segments pair verifies the following condition:

$$dist(A_j B_j, A_{j+1} B_{j+1}) \geq R_j + R_{j+1} \quad (4.16)$$

Figure 17 – Distance between two segments.



Source: (KELAIAIA; ZAATRI et al., 2012).

To avoid cranks and limbs collision during movement this work uses as reference the radius of Ceart's 6-RUS limbs, which means at least distance of $3.1cm$ for cranks and $3cm$ for passive link.

4.2 Corresponding eigenvalue problem

In order to solve the constraint optimization problem of Equation 4.2, it is transformed into an unconstrained optimization problem. This transformation is achieved by forming the Lagrangian. $L(\$, \lambda) = F(\$) - \lambda h(\$)$ and solving the resulting unconstrained optimization problem.

$$\min L(\$, \lambda) \quad (4.17)$$

where for this particular problem, the Lagrangian becomes

$$L(\$, \lambda) = \$^T M \$ - \lambda (\$^T D \$ - c) \quad (4.18)$$

The minimization of the Lagrangian is now performed by taking the partial derivative of the Lagrangian with respect to λ and $\$$ and setting both equations equal to zero. With this, all extrema of the problem are identified which, since F is bounded from below by 0, must include the absolute minimum of F . Differentiating the Lagrangian with respect to λ results in

$$\frac{\partial L}{\partial \lambda} = (\$^T D \$ - c) = 0 \quad (4.19)$$

which is the constraint from the Equation (5.10). Differentiating the Lagrangian with respect to $\$$ and using the fact that M and D are symmetric yields:

$$\frac{\partial L}{\partial \$} = 2M\$ - 2\lambda D\$ = (M - \lambda D)\$ = 0 \quad (4.20)$$

For a non trivial solution to exist, the matrix expression in the parentheses must be singular. In other words:

$$\det(M - \lambda D) = 0 \quad (4.21)$$

which is called the corresponding general eigenvalue problem. From this, the eigenvalues, λ , (*i.e.* the stationary points in a minimization sense) and the associate eigenvector, $\$$, are computed. The eigenvectors are then scaled to satisfy the constraints, $\$^T D \$ - c = 0$, and the resulting scaled vectors are substituted into the original function, F , from Equation 4.10 to yield its minimum value.

Furthermore, the objective function's minimum is the smallest eigenvalue, λ_{min} , of Equation 4.21 multiplied by the constant c . This is proven by first going back to Equation 4.20 which rewritten yields

$$M \$ = \lambda D \tag{4.22}$$

Substituting this relationship into objective function F and using the constraint g yields.

$$F(\$) = \$^T E \$ = \lambda \$^T D \$ = c\lambda \tag{4.23}$$

Since c is a positive constant, F is minimised for the smallest eigenvalue, λ_{min} , thus.

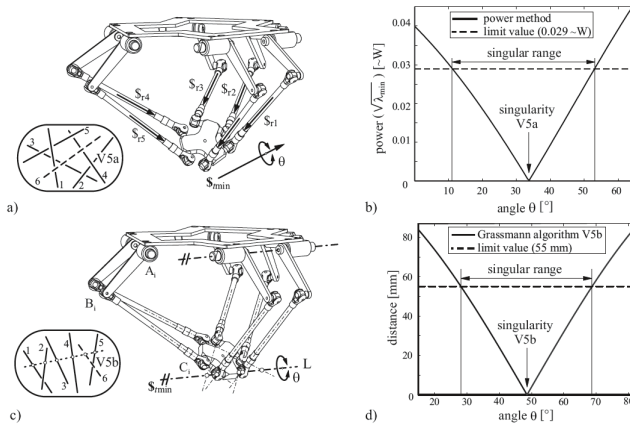
$$\min F(\$) = c\lambda_{min} \tag{4.24}$$

Another result that can be inferred from this analysis is all eigenvalues are non-negative, $\lambda_i \geq 0$, since the objective function is non-negative.

Then $\sqrt{\lambda_{min}}$ is associated to the minimum power [W] of the system, which indicates the manipulator singularity closeness. The smallest eigenvalue λ_{min} will be the minimum value of the objective function $F(\$)$, and so it can be utilized as a measure value.

The measure behaviors of the minimum power of the system through a singularity is showed in Figure 18, here the end-effector twists θ^o around the S_{min} (the end-effector twist which requires minimum power in this singularity). The singularity occurs when $\sqrt{\lambda_{min}} = 0$, but a singular range exists due to clearance and compliance of the system, where the end-effector is still unconstrained. Theoretically the index value is supposed to be zero, but due joints clearance the value of the index to be a singularity is slightly higher (LAST et al., 2005). The singular range bound is experimentally identified as 0.029 W (IWF-Hexa Robot) and upon it the manipulator stiffness is warranted into the whole workspace (HESSELBACH et al.,). The index minimum value is determined by evaluating the built robot prototype belonging IWF-robot, this work suppose that CEART-robot have a similar value, for the optimization.

Figure 18 – a) Grassmann variety V5a on the Hexa; b) Power based index; c) Grassmann variety V5b; d) Grassman V5b based index.



Source: (HESSELBACH et al.,).

The power inspired measure has two weakness. The first one comes

from the normalisation procedure, namely the fact that the angular part of the twist is constrained to have unit norm. Thus singularities with a purely translational deficiency are not detected by the measure. Pottman *et al.* propose to supplement this approach with second optimisation for which the twist is normalized to have unit linear part. However, this results in two different values for any configuration and it is unclear how these two values are to be combined to yield a single for closeness to singularity.

5 OPTIMIZATION

Optimization is the act of obtaining the best result under given circumstances. In design, construction, and maintenance of any engineering system, engineers have to take many technological and managerial decisions at several stages. The ultimate goal of all such decisions is either to minimize the effort required or to maximize the desired benefit (RAO; RAO, 2009).

This chapter describes optimization methods which were used in this work to maximizing the orientation rotation avoiding the direct singularity configuration and others constraints. Two optimization methods were employed in this work. The particle swarm optimization PSO was applied first to escape for local minimum problems during the optimization. Then Matlab's *fmincon* function work to guarantee the optimal solution since is set with Intern point algorithm which is a gradient based method. This work used the PSO Matlab codes developed by (DONCKELS, 2006).

5.1 Particle Swarm Optimization

The particle swarm optimization (PSO) is a technique introduced by James Kennedy and Russell Eberhart in the 90's and emerged from experiments with algorithms that model the *social behavior* observed in some species of birds (KENNEDY; KENNEDY; EBERHART, 2001).

PSO is a robust stochastic optimization method based upon the behavior of swarms observed in nature. The method captures the concept of social intelligence and co-operation. The PSO method employs particles that exhibit individual and group behavior when looking for a solution within a

search space. These particles also have a memory of where they have been.

PSO optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search space according to simple mathematical formulas on the particle position and velocity. The motion of each particle is influenced by its best known site position, but is also oriented towards the most known locations in the search space, which are updated as better positions are found by other particles. This is expected to move the swarm to the best solutions (KENNEDY; KENNEDY; EBERHART, 2001).

PSO is a metaheuristic technique, as it makes little or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, as PSO metaheuristics do not guarantee an optimal solution is always found. More specifically, the PSO does not use the gradient of the problem to be optimized (KENNEDY; KENNEDY; EBERHART, 2001).

5.1.1 Particle Swarm Optimization with constriction factor

PSO model consists of a swarm of particles, which are initialized with a population of random candidate solutions. In 2002, Clerc and Kennedy introduced a constriction factor in PSO, which was later on shown to be superior to the inertia factors (CLERC; KENNEDY, 2002). They move iteratively through the d -dimension problem space to search the new solutions, where the fitness, f , can be calculated as the certain qualities measure. Each particle has a position presented by a position-vector V_i (i is the index of particle), and a velocity represented by a velocity-vector V_i . Each particle remembers its own

best position so far in a vector P_i^{best} , and its d -th dimensional value is P_{id}^{best} .

The best position-vector among the swarm so far is then stored in a vector G^{best} , and its d -th dimensional value is G_d^{best} . During the iteration time k , the update of velocity from the previous velocity to the new velocity is determined by Equation 5.1. The new position is then determined by the sum of previous position and the new velocity by Equation 5.2.

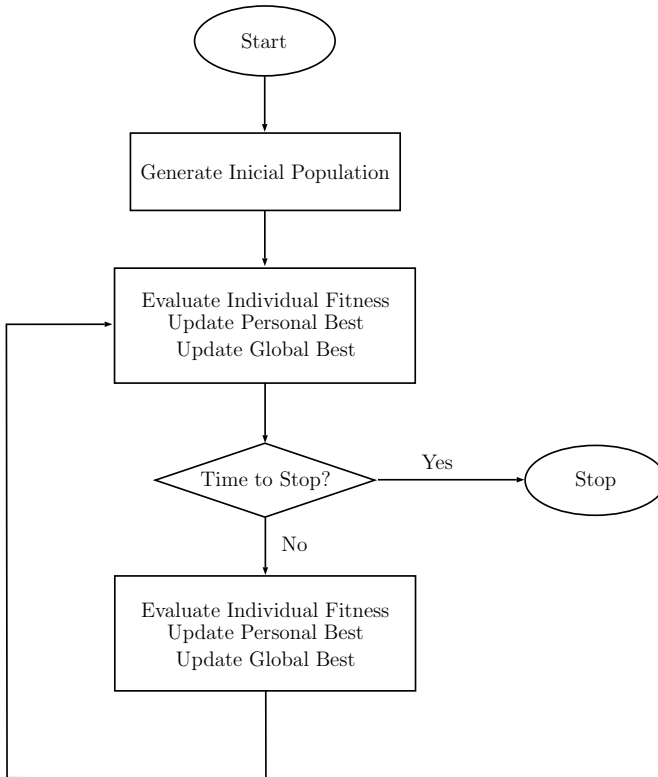
$$V_{id}(k+1) = K[V_{id}(k) + c_1 r_1 (P_{id}^{best}(k) - P_{id}(k)) + c_2 r_2 (G_d^{best}(k) - P_{id}(k))] \quad (5.1)$$

$$P_{id}(k+1) = V_{id}(k+1) + P_{id}(k) \quad (5.2)$$

$$K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \text{ where } \varphi = c_1 + c_2, \varphi > 4 \quad (5.3)$$

In Equations 5.1-5.3, φ is called constriction factor, r_1 and r_2 are random numbers, which are used to maintain the diversity of population, and are uniformly distributed in the interval $[0, 1]$ for the d -th dimension of i -th particle. c_1 is a positive constant, called as coefficient of self-recognition component, c_2 is a positive constant, called as coefficient of social component. From Equation 5.1, a particle decides where to move next, considering its own experience, which is the memory of its best past position, and the experience of its most successful particle in the swarm (see Figure 19).

Figure 19 – Flowchart for particle swarm optimization algorithm.



Source: (KACHITVICHYANUKUL, 2012).

5.1.2 Configuration for Particle Swarm Optimization

In order to obtain a good performance of the PSO when solving the optimization problem some configurations are necessary. As previously mentioned, the PSO has two coefficients, called coefficient of self-recognition component and coefficient of social component. These parameters determinate the particle movement influence from its own experience and from the most successful particle in the swarm. As recommended by Schutte and Groenwold

Table 2 – PSO coefficients values.

Symbol	PSO Coefficient and Parameters	Value
c_1	<i>coefficient of self-recognition component</i>	2.8
c_2	<i>coefficient of social component</i>	1.3
φ	<i>constriction factor</i>	4.1
K	<i>constant multiplier</i>	0.7298

Source: Own author.

(SCHUTTE; GROENWOLD, 2005), the coefficient of self-recognition and coefficient of social component have their values as shown in Table 2. In this case, constriction factor, φ was set to 4.1 and the constant multiplier K is thus 0.729.

$$\varphi = c_1 + c_2 = 2.8 + 1.3 = 4.1, \text{ where } \varphi > 4 \quad (5.4)$$

$$K = \frac{2}{|2 - 4.1 - \sqrt{(4.1)^2 - 4 \cdot 4.1}|} = \frac{2}{|-2.7403|} = \frac{2}{2.7403} = 0.7298 \quad (5.5)$$

The Swarm Size which means number of particles and the max number of generation are other parameters to configure in PSO algorithm. The Swarm Size directly influences the algorithm performance.

5.2 FMINCON

Fmincon is a Matlab function part of Matlab's Optimization Toolbox. It's find a minimum of a constrained nonlinear multivariable function. *fmincon* attempts to find a constrained minimum of a scalar function of several variables starting at an initial estimate. *fmincon* is a gradient-based method that is

designed to work on problems where the objective and constraint functions are both continuous and have continuous first derivatives.

There is no guarantee that the *fmincon* will return a global minimum, unless the global minimum is the only minimum. As a method based on gradient may get remand to a local minimum and not find the value of the global minimum. Based on where *fmincon* starts, it may terminate at the global minimum or at one of the local minimum. The optimizer gets trapped in the local valley and can't escape to reach the global valley. The FMINCON function, configured to use the Interior Point algorithm is employed to accelerate the search for bottom of the valley indicated by the PSO as the best particles locations.

5.2.1 Interior Point Algorithm

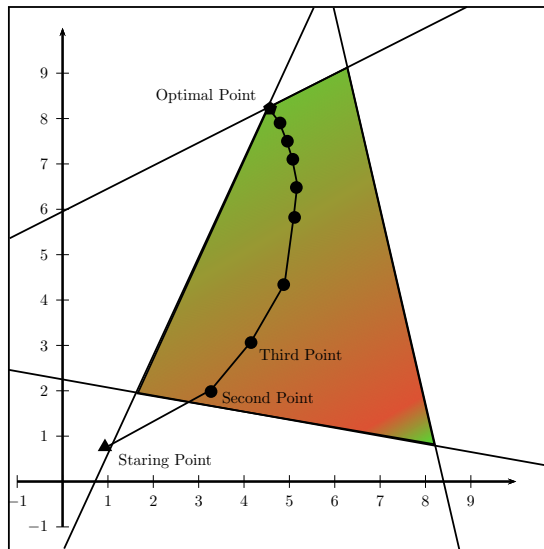
An Interior Point method is a linear or nonlinear programming method that achieves optimization by going through the middle of the region defined by the problem rather than around its surface (FORSGREN; GILL; WRIGHT, 2002).

A polynomial time linear programming algorithm using an interior point method was developed by (KARMAKAR, 1984). Arguably, interior point methods were known as early as the 1960s in the form of barrier function methods, but the media hype accompanying Karmarkar's announcement led to these methods receiving a great deal of attention. However, it should be noted that while Karmarkar claimed that his implementation was much more efficient than the simplex method, the potential of interior point method was established only later. By 1994, there were more than 1300 published papers

on interior point methods.

Current efficient implementations of interior methods or are mostly based on a predictor-corrector technique (MEHROTRA, 1992), where the Cholesky decomposition of the normal equation or LDL^T factorization of the symmetric indefinite system augmented system is used to perform Newtons method (together with some heuristics to estimate the penalty parameter). All current interior point methods implementations rely heavily on very efficient code for factoring sparse symmetric matrices.

Figure 20 – Internal Point Algorithm.



Source: Own author.

In an interior point method, each iteration considers points that are strictly inside the feasible region, with the exception of the initial point which may be outside the feasible region. In each iteration a new point $x' = x + \alpha D_x$ is computed from the current point x by finding a step direction D_x . The step

direction is chosen to optimize the change in objective value from x to x' . The parameter a is used to ensure that the new point x' lies strictly inside the feasible region.

5.2.2 Sensitivity Analysis

In numerical methods of optimization, one must determine the effect of a change in the current design on the cost and constraint function i.e. evaluate the gradient of response quantities with respect to design variable. This is commonly known as design sensitivity analysis and can constitute a major task in any structural optimization program. The gradient also important in their own right as they represent trend for the structural performance or constraint function. The derivatives of the cost and constraint function are essential to compute a search direction in the optimization process.

The finite difference approximations for derivatives are one of the simplest and of the oldest methods to solve differential equations. It was already known by L. Euler (1707-1783) ca. 1768, in one dimension of space and was probably extended to dimension two by C. Runge (1856-1927) ca. 1908. The advent of finite difference techniques in numerical applications began in the early 1950s and their development was stimulated by the emergence of computers that offered a convenient framework for dealing with complex problems of science and technology. Theoretical results have been obtained during the last five decades regarding the accuracy, stability and convergence of the finite difference method for partial differential equations.

The principle of finite difference methods is close to the numerical schemes used to solve ordinary differential equations. It consists in approxi-

mating the differential operator by replacing the derivatives in the equation using differential quotients. The domain is partitioned in space and in time and approximations of the solution are computed at the space or time points. The error between the numerical solution and the exact solution is determined by the error that is committed by going from a differential operator to a difference operator. This error is called the discretization error or truncation error. The term truncation error reflects the fact that a finite part of a Taylor series is used in the approximation.

For the sake of simplicity, we shall consider the one-dimensional case only. The main concept behind any finite difference scheme is related to the definition of the derivative of a smooth function u at a point $x \in \mathbb{R}$

$$u'(x) = \lim_{h \rightarrow 0} \frac{u(x+h) - u(x)}{h} \quad (5.6)$$

and to the fact that when h tends to 0 (without vanishing), the quotient on the right-hand side provides a "good" approximation of the derivative. In other words, h should be sufficiently small to get a good approximation. It remains to indicate what exactly is a good approximation, in what sense. Actually, the approximation is good when the error committed in this approximation (i.e. when replacing the derivative by the differential quotient) tends towards zero when h tends to zero. If the function u is sufficiently smooth in the neighborhood of x , it is possible to quantify this error using a Taylor expansion.

5.3 Hybrid Optimization

Finding a global optimal solution is a challenging task in many applications due to the fact that data and models are usually nonlinear and subject to different sources of error. In many cases the number of these parameters may be significantly large since they generally depend on a prescribed level of spatial and temporal resolution. Most successful approaches in global optimization use some stochastic or heuristic strategies that do not depend on derivative information, e.g., genetic algorithms, tabu searches, neural networks, and simulated annealing, among others (MENEZES; OORSCHOT; VANSTONE, 1996). Despite inherent costs, their success resides in making a broad and systematic exploration of the search domain, although this may vary with the nature of the problem. Therefore, it is useful to obtain several local optimal solutions and compare them before selecting a global one. However, the performance of these algorithms depends significantly on the choice of parameters and this task tends to be less obvious on large scale since the number of function evaluations grows dramatically with the number of parameters. On the other hand, the availability of derivative information, such as gradient, and Jacobian or Hessian operators, allows for achieving higher rates of convergence towards a local optimum.

One way to achieve these goals is through the coupling of two or more different optimization algorithms, which have complementary characteristics, which results in so-called hybrid algorithms. It is common to find hybrid algorithms involving an algorithm of type heuristic used to cover the entire search space and identify the region where the global minimum is found, and

an algorithm with mathematical reasoning, said non linear programming, able to quickly reach the minimum, since the region has been identified. This type of strategy improves reliability compared to methods of non linear programming, because it is more likely to find the global minimum, and increases efficiency compared with pure heuristic algorithms (WANG; ZHENG, 2001).

As the optimization problem of this work has many geometrical parameters as input arguments, therefore an ample search space and certainly several local minima, a hybrid approach is proposed. First, the PSO is employed for exploring the problem space and avoiding getting stuck in local minima. After the PSO finds the neighbourhood of the optimum, an interior point algorithm is adopted for exploiting the valley and accelerate the search for the bottom of it. But one important note is that the optimum found by Interior Point may not be the global optimum because there is no guarantee PSO algorithm find the global valley.

6 ORIENTATION WORKSPACE OPTIMIZATION FOR A 6-RUS PARALLEL ROBOT

This work presents a method determine active joint optimal location (position and orientation) for a flight simulator based on a 6 - RUS parallel robot aiming at maximize its orientation workspace.

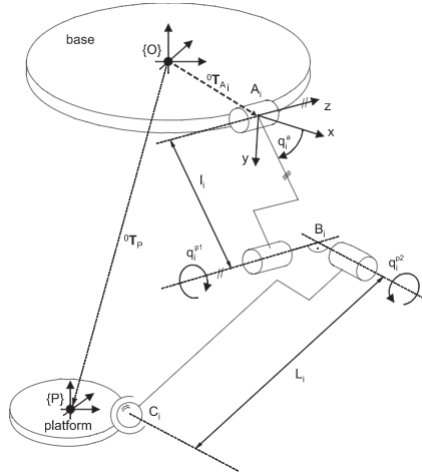
During the development of this work, the inverse kinematic of CEART-robot was determined (International Congress of Mechanical Engineering), (CAMPOS et al., 2014). As by product of this dissertation, an additional article was produced and published in CONEM 2014 (Congresso Nacional de Engenharia Mecânica), when the Matlab program provided some preliminary results, (FURTADO; CAMPOS; REIS, 2014).

6.1 Kinematical Performance Index for Two 6 - RUS Parallel Robots

The Six DOF IWF-robot is composed by six limbs connecting the basis to the end-effector (see Figure 21). Each limb contains an active rotative joint A_i (for $i = 1, \dots, 6$) fixed to the basis, a passive universal joint B_i (composed by two orthogonal rotative joints) and a passive spherical joint C_i connected to the end-effector. The cranks and the passive links are connected at B_i .

As the IWF-robot and the CEART-robot robots that have same kinematic chain and therefore belong to 6-RUS parallel manipulator family, it is possible to compare their performance in the same task and study the geometric differences and their influence on performance. The Ceart CEART-robot

Figure 21 – 6-Rus Kinematic Chain.



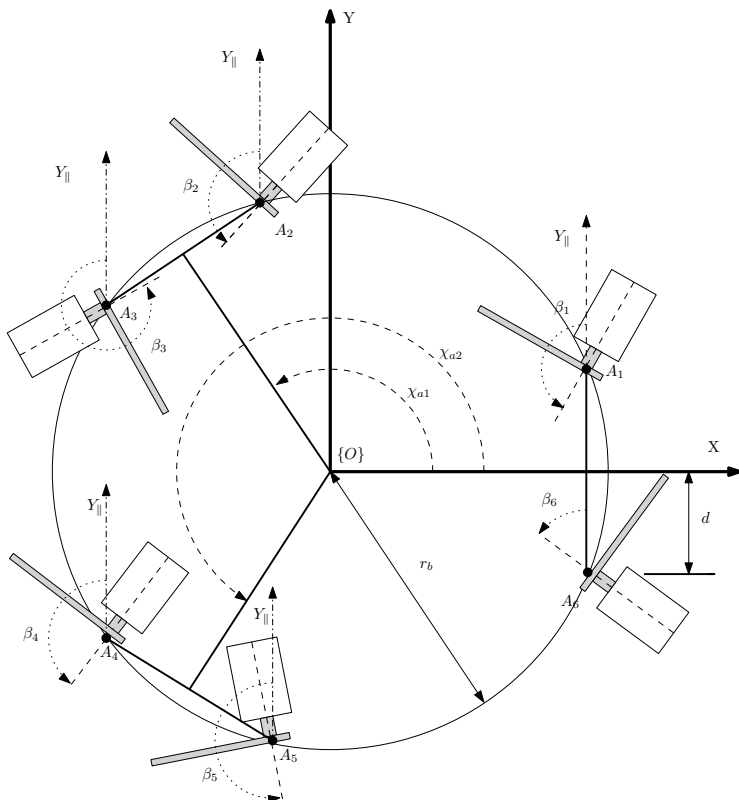
Source: (LAST et al., 2005).

was conceived to result in a complete operating flight simulator built by the university UDESC(Universidade do Estado De Santa Catarina. Hence both configurations should be tested with a typical flight simulation task.

Based in the inverse kinematics presented in Chapter 2 and looking for the differences between IWF-robot and CEART-robot, a better understanding of 6-RUS robots and their geometrical parameters are required.

The angle β_i refers to the Y_{mi} ($i = 1, \dots, 6$) axis (active joint coordinates system) to $Y_{||}$ (a parallel axis to Y of overall system), and it determines the active joint orientation (see Figure 22). With d (half-distance between actuated joint pair: 1 – 6, 2 – 3 and 4 – 5) and r_b (base radius) constants; χ_{a1} and χ_{a2} that the active joint location. Each active joint has a local coordinates system with origin at O_{mi} ($i = 1, \dots, 6$) (see Figure 23).

Figure 22 – Base layout, where design parameters are χ_{a1} , χ_{a2} , β_1 , β_2 , β_3 , β_4 , β_5 and β_6 .

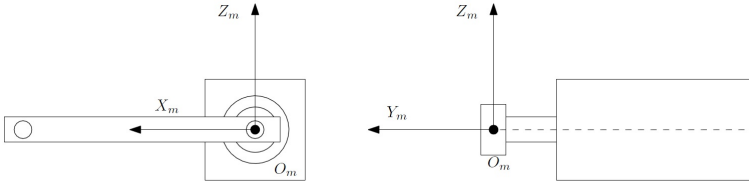


Source: Own author.

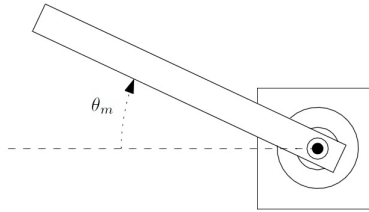
This system location determines the active joint situation. Active joints system origin O_{mi} are the A_i points from inverse kinematic calculus. Active joint rotation angle θ is zero in the crank horizontal position as shown in Figure 24.

6-RUS parallel architecture may be considered as a parallel robots family. Each member of this family has different values for β_i angles. CEART-

Figure 23 – Active joint local system.



Source: Own author.

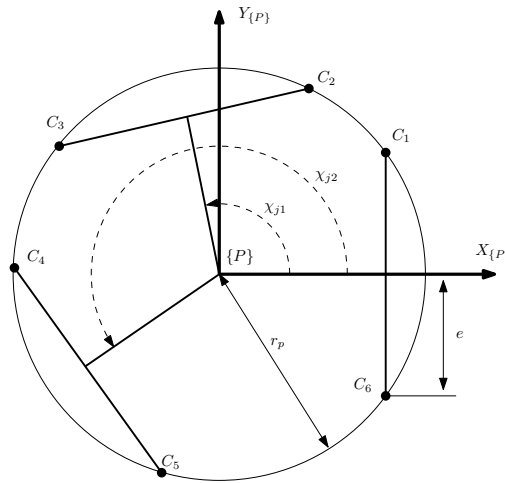
Figure 24 – Active Joint Angular Position θ .

Source: Own author.

robot is a member of this family with $\beta_1 = 120$, $\beta_2 = 180$, $\beta_3 = 240$, $\beta_4 = 300$, $\beta_5 = 0$ and $\beta_6 = 60$, and IWF-robot is another member with $\beta_1 = 0$, $\beta_2 = 120$, $\beta_3 = 120$, $\beta_4 = 240$, $\beta_5 = 240$ and $\beta_6 = 0$.

Let be considered constraint e (half-distance between pair of joints) and r_p (end-effector radius) constants; therefore knowing χ_{j1} and χ_{j2} , the points C_i may be determinated. These points are the spherical joints location for each limb, i.e., the passive link connection with the end-effector (see Figure 25).

Figure 25 – Platform angular layout, where the design parameters are χ_{j1} , χ_{j2} , r_p and e .



Source: Own author.

Table 3 – IWF-robot angles parameters.

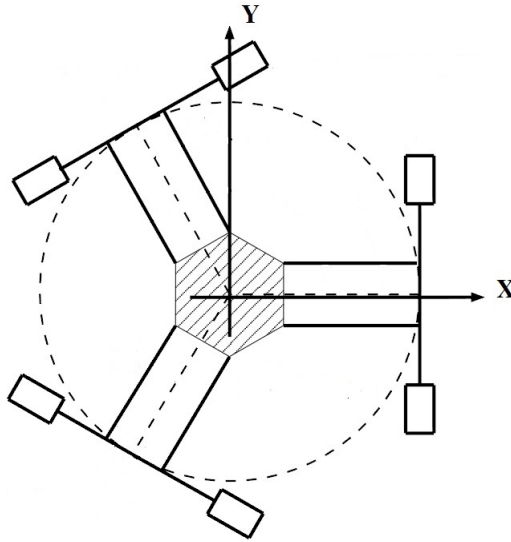
Symbol	1	2	3	4	5	6	Geometric Parameter
χ_a	0	120	120	240	240	0	APact (Base)
χ_j	0	120	120	240	240	0	APjoint (End-Effector)
β	0	120	120	240	240	0	aact (Base)

Source: Own author.

6.2 Geometrical Parameters of the IWF-robot and CEART-robot

Methods and strategies presented in this paper are implemented and tested in a 6-RUS IWF-robot. The six limbs of IWF-robot are arranged in three pairs, each pair contains two active joints whose, i.e. drive axes, are collinear (see Figure 26).

Figure 26 – IWF-robot active joint location.



Source: (CAMPOS et al., 2011).

Table 4 – IWF-robot Geometrical Parameters.

Symbol	Value	Geometric Parameter
r_b	0.360	Base Radius
r_p	0.05196	Platform Radius
r_i	0.240	Length of Crank Link
R_i	0.564	Length of Passive Link
e	0.0516	HDBjoint (End-Effector)
d	0.0516	HDBact (Base)

Source: Own author.

IWF-robot have the geometrical parameters as in Table 3. Other geometrical parameters necessary to define the kinematic structure of the IWF-robot are presents in Table 4:

For the geometrical parameters of CEART-robot see Table 5. Other geometrical parameters necessary to define the kinematic structure of the this

Table 5 – CEART-robot angles parameters.

Symbol	1	2	3	4	5	6	Geometric Parameter
χ_a	0	120	120	240	240	0	APact (Base)
χ_j	0	120	120	240	240	0	APjoint (End-Effector)
β	120	180	240	300	0	60	aact (Base)

Source: Own author.

Table 6 – CEART-robot Geometrical Parameters.

Symbol	Value	Geometric Parameter
r_b	0.636885	Base Radius
r_p	0.29173	end-effector Radius
l_i	0.194	Length of Crank Link
L_i	0.650	Length of Passive Link
e	0.197555	HDBjoint (End-Effector)
d	0.402345	HDBact (Base)

Source: Own author.

robot are shown in Table 6.

6.3 IWF-robot and CEART-robot Performance for Flight simulator Task

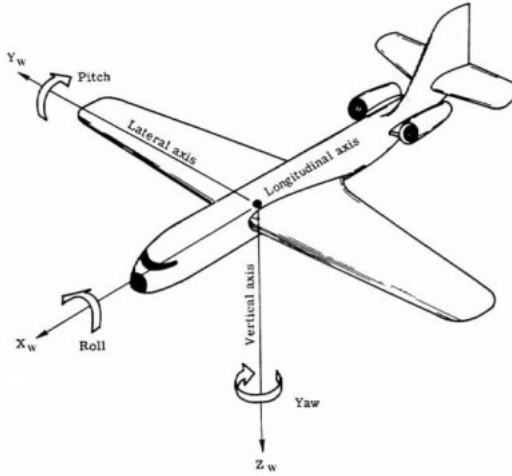
The IWF-robot and CEART-robot may have just a few kinematic parameter values as difference, but they also have distinction of performance. From this, a question arises, what setting has the best performance for a flight simulation task?

6.3.1 Proposed task

An aircraft in flight is free to rotate in three dimensions: *pitch*, nose up or down about an axis running from wing to wing; *yaw*, nose left or right

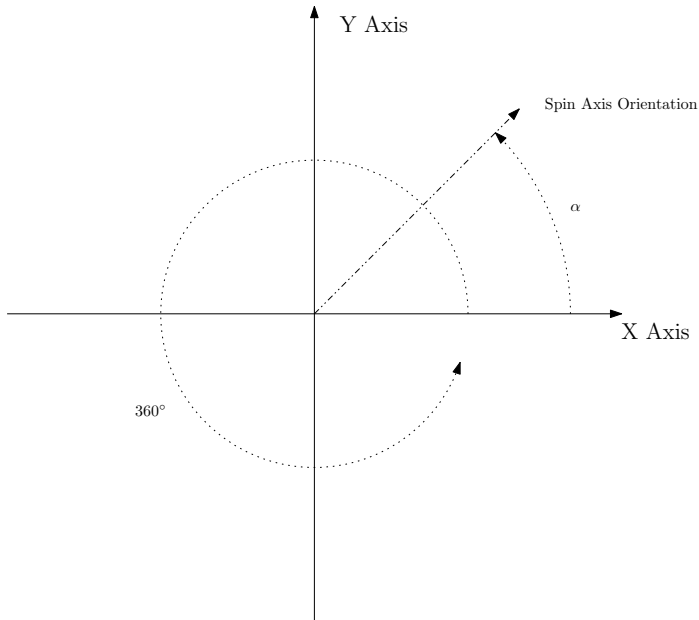
about an axis running up and down; and *roll*, rotation about an axis running from nose to tail. The axes are alternatively designated as lateral, vertical, and longitudinal (see Figure 27).

Figure 27 – Pitch-Roll-Yaw of an Air Plane



Source: Learn To Fly Drones. Available: <http://www.dronethusiast.com/learn-fly-drones-newbies-guide-useful-resources/> Accessed: June 8, 2015.

The movement executed by the manipulator is a orientation rotation in the initial position. As the orientation rotations in Z axis can be easily performed by a additional motor, the optimization should be applied in orientations in XY plane. For a given χ_a and β_i angles (optimization parameter), a rotation is performed around all orientation axes in the plane XY (see figure 28) to verify where the rotation is minimum and its value.

Figure 28 – Axis Orientation Rotation with α from 0° to 360° 

Source: Own author.

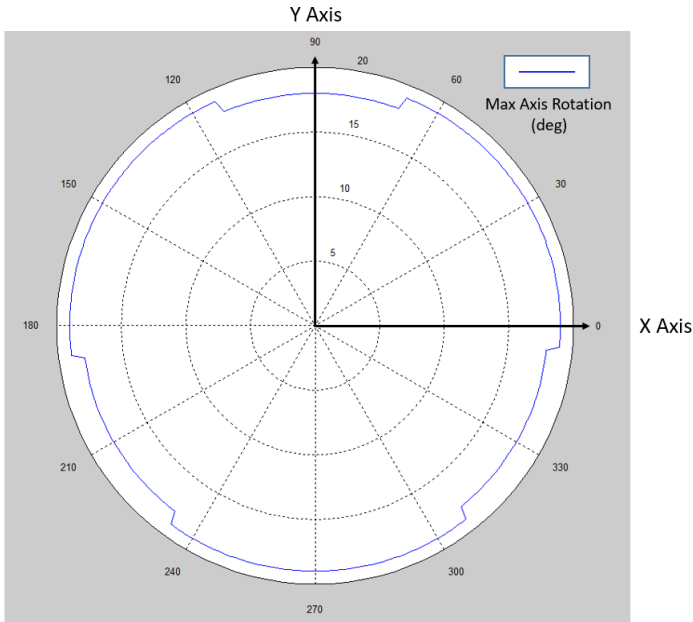
6.3.2 IWF-robot and CEART-robot PERFORMANCE

In order to illustrate IWF-robot and CEART-robot maximum rotation around 360° axes in XY plane, a polar graphics is made for each configuration. The inner circle represents the maximum rotation that the manipulator may perform in all orientation axes in the XY plane from 0° to 360° .

As figures 29 and 30 show, CEART-robot have a better performance than IWF-robot in this movement. This optimization method intends to find a 6-RUS configurations with a better result in this movement, i.e., with maximum rotation around all XY axes.

If we have a closer look in these configurations it is possible to find

Figure 29 – IWF-robot Polar Graphic.



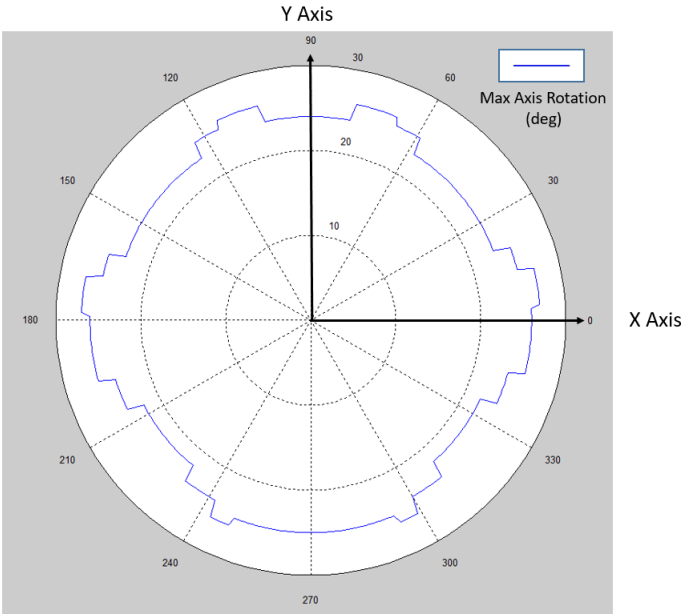
Source: Own author.

deficiencies of each configuration. Therefore, using a kinematic algorithm, a new task consisting of performing rotations around one single axis in order to evaluate the kinematic index performance, can be done.

For instance IWF-robot with respect to Y axis, is shown in Figure 31. It is observed that it fails the minimum work index value in 18° of rotation in this axis. Which is the same angle present in Polar Graphic. IWF-robot configuration typically have direct singularity issues.

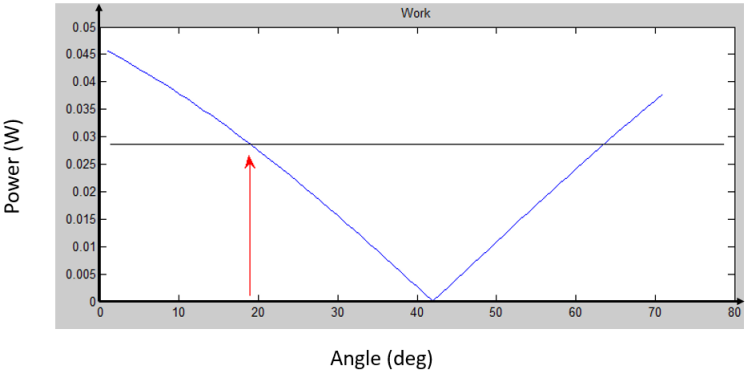
Figure 32 confirms this showing the IWF-robot was not near the inverse singularity, as defined in Section 4.1.3.

Figure 30 – CEART-robot Polar Graphic.



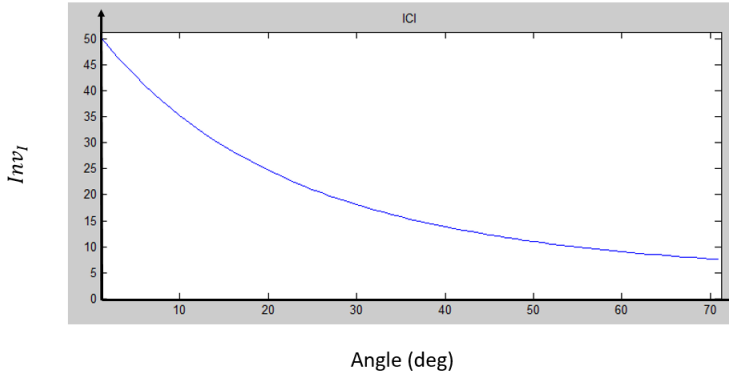
Source: Own author.

Figure 31 – IWF-robot direct singularity index, see red arrow.



Source: Own author.

Figure 32 – IWF-robot inverse singularity constraint.



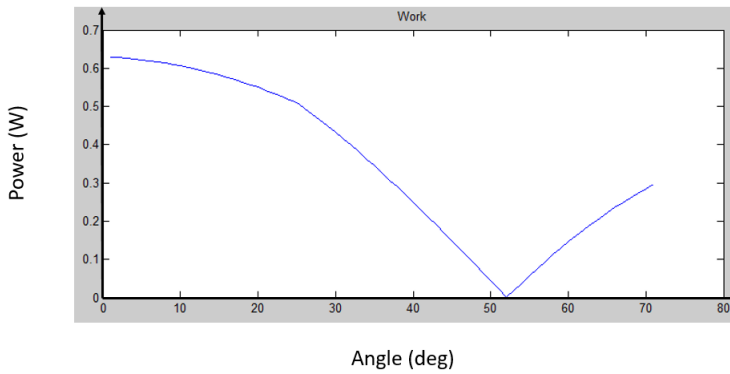
Source: Own author.

The CEART-robot performance with respect to rotation around the Y axis, see (Figure 33), shows that it didn't fail in the work index until more than 30° of rotation in this axis. However the polar graphic has a smaller rotation value, therefore it fails in the inverse singularity, see (Figure 34). CEART-robot had a limited movement because the inverse singularity, shows up for 25° in this case, as Section 4.1.3. defined the inverse singularity problem.

6.4 6-RUS Robot Optimization for Flight Simulator Task

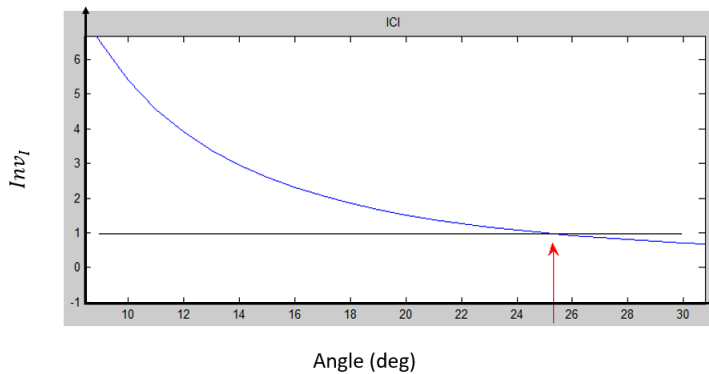
In this work, a MATLAB algorithm is developed in order to optimize a 6-RUS Robot. The selected parameters to optimize the orientation workspace are the angles $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6$ and d . These kinematics parameters define the active joint locations. Other design parameters to optimize are; half of the distance between a pair of spherical joints in the end-effector, e ; radius of the end-effector, r_p and the end-effector height, which is the end-effector Z

Figure 33 – CEART-robot direct singularity index.



Source: Own author.

Figure 34 – CEART-robot inverse singularity constraint, see red arrow.



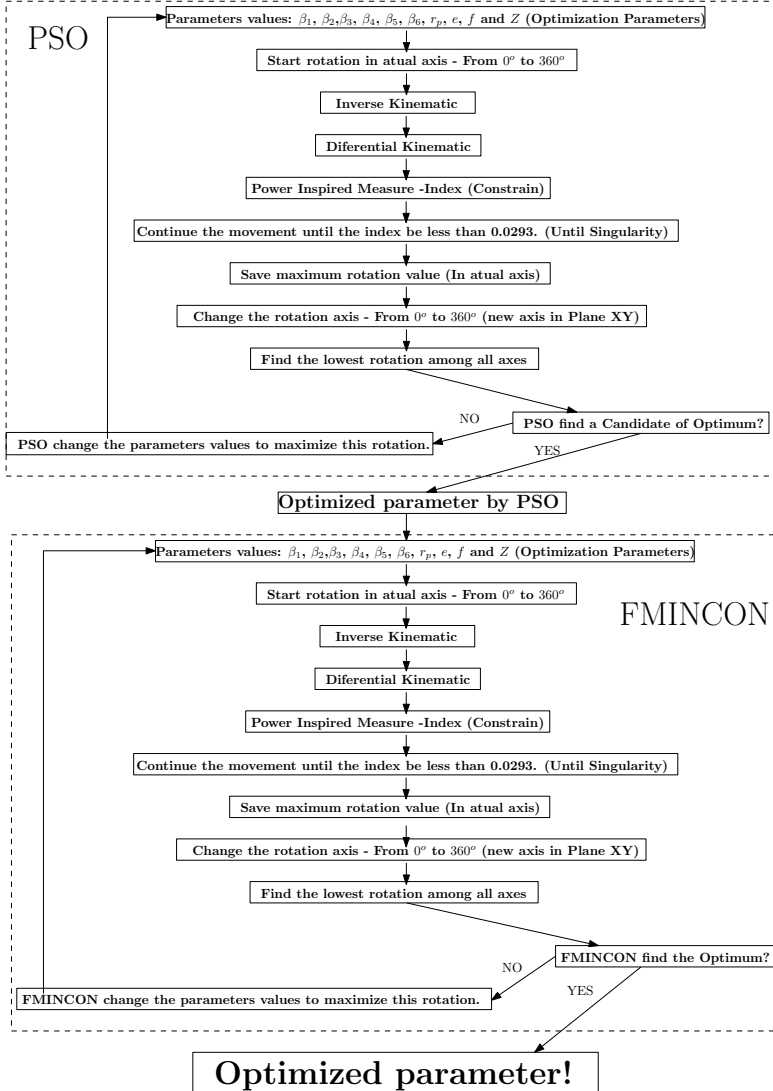
Source: Own author.

position.

The program initially receives a geometrical parameters to optimize (optimization parameters) and execute the robot movement simulation through inverse kinematic. The screw theory is applied to establish the wrenches and twist in order to determined the singularity index that is the system constraint.

As shown in section 4.2, the index can not have values below 0.0293 which determines a singular position.

Figure 35 – Flowchart of MATLAB program optimization program.



Source: Own author.

The algorithm calculates the power inspired measure, aiming to evaluate the closeness to a direct singularity. This algorithm minimizes the power of the six wrenches (through the lines $\overline{B_i C_i}$, $i = 1, \dots, 6$) over the less constrained end-effector twist, for a given configuration.

This algorithm takes the instantaneous position for points B , and C , to build the wrenches acting upon the end-effector, see (Figure 14). For convenience, it is choose the wrenches with magnitude $\tau = 1$. Given that only spherical joints connect the passive links with the end-effector, only pure forces are transmitted to the end-effector, *i.e.* null pitch wrenches with $\overline{C_i B_i}$ directions.

The Graminiam matrix M is formed from the normalized wrenches $\$'_i$ Equation 2.5 and then the eigenvalue problem is stated by Equation 2.12. A minimum power function $\sqrt{\lambda_{min}}$ and the less constrained end-effector twist $\$_{min}$ (axis and pitch) are calculated while the end-effector is moving. Theoretically, when the manipulator is at a singular pose, the end-effector has only a very small instantaneous motion. However, in practice, all manipulators have some amount of clearances (and similarly some compliance) and allow some finite motion at the end-effector. This motion is denoted as the unconstrained end-effector motion. The power based measure $\sqrt{\lambda_{min}}$ indicates how much unconstrained end-effector motion is allowed.

The optimization algorithms evaluate the index value to adjust β angles value, e , f , r_p and end-effector Z position to maximize the rotation, (positives and negatives) around axis in the plane XY , evaluating the index value. The program starts with an initial estimation of the angles. Then

maximum value of these rotations is defined, for this configuration, until the movement produced the minimum index value.

After this loop, the PSO algorithm is applied to estimate new angles with better rotations values. This is the applied optimization algorithm to escape from local minimum. PSO tries to find the global minimum until no better values are found. Then it sends the resultant design parameters for initial estimation to Internal Point Algorithm. Then Internal Point starts with design parameters found by PSO and establishes the optimum, see (Figure 35).

6.5 Optimization Result and Kinematic Analysis

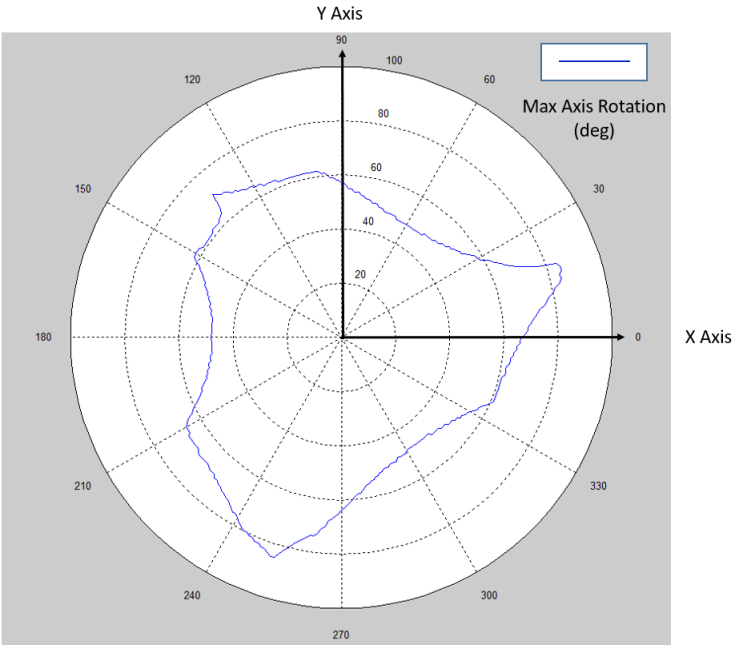
This section presents the optimization result performed by the proposed program. Some kinematic analysis may be made from 6-RUS optimized configuration using mathematical software e.g. Matlab and with a ADAMS CAE model.

6.5.1 Software Tool: Matlab

Aiming at a better orientation workspace, optimization method is applied for 6-RUS robot by Matlab algorithm which reconfigure some geometrical parameters. The geometrical parameters to optimize the orientation workspace are β_i angles, e , f , r_p and Z position (see Figures 22 and 25).

A MATLAB (MATrix LABoratory) program to perform the calculations and apply this optimization is developed. MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming.

Figure 36 – Optimized 6-RUS Polar Graphic.



Source: Own author.

Table 7 – Active joints orientation angle for 6-RUS optimized robot.

Symbol	1	2	3	4	5	6	Geometric Parameter
β	60	159	181	285	302	44	aact (Base)

Source: Own author.

The Z position for optimized orientation workspace is $Z = 0.4042$. Table 7 presents β angles values and Table 8 the geometrical parameters. Polar graphics for 6-RUS optimized configuration, shows that the robot may perform at least 47.5° degrees rotations in all orientation axes in the XY plane.

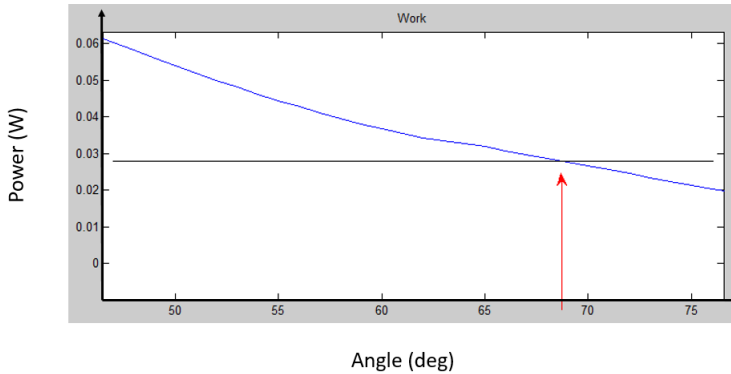
Aiming at get closer look to this configurations just rotate around one axis to evaluate the kinematic index performance. Optimized 6-RUS

Table 8 – 6-RUS optimized parameters.

Symbol	Value	Geometric Parameter
r_p	0.1972	End-Effector Radius
e	0.0766	HDBjoint (End-Effector)
d	0.0501	HDBact (Base)

Source: Own author.

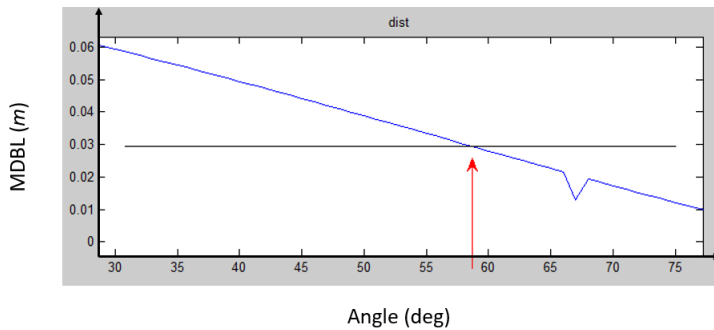
performance in rotation in Y axis shows that the minimum work index value remains suitable until 60° of rotation around this axis, see (Figure 37), however polar graphic has a max rotation of 57° in this axis, so it fails due in some constraint.

Figure 37 – Direct singularity index for Y rotation, see red arrow.

Source: Own author.

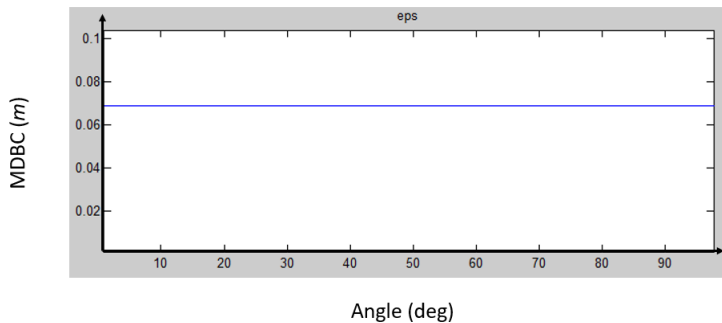
Considering the 6-RUS optimized robot movement around Y axis, it is possible to observe that some pair of limbs, about 57° collided, see Figure 38 for the minimum distance between any pair of limbs (MDBL), therefore the movement limited due limbs proximity/collision. See Figure 39 to verify the minimum distance between any pair of cranks (MDBC), remain constant

Figure 38 – Passive links minimum distance.



Source: Own author.

Figure 39 – Cranks minimum distance, see red arrow.

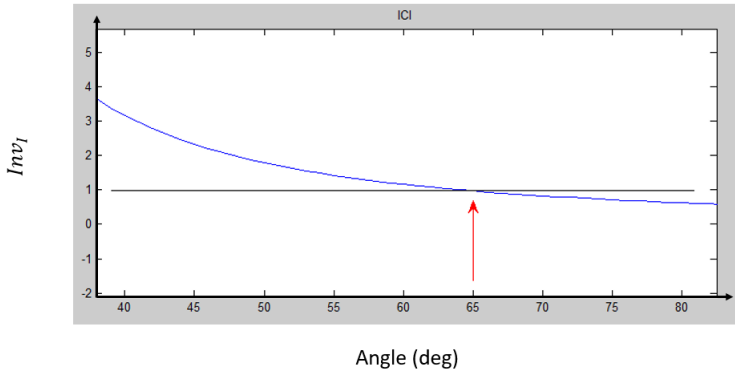


Source: Own author.

i.e. the cranks do not have any problem with collision due this movement.

Distance between cranks d distance remains constant. Another constraint to analyze is the inverse singularity proximity index, which fails about 65° , meaning that some limb or limbs are fully extended or contracted. Therefore some discontinuity in the crank movement may be seen, see (Figure 41 and 42). In the neighborhood of 65° the cranks movement has a discontinuity

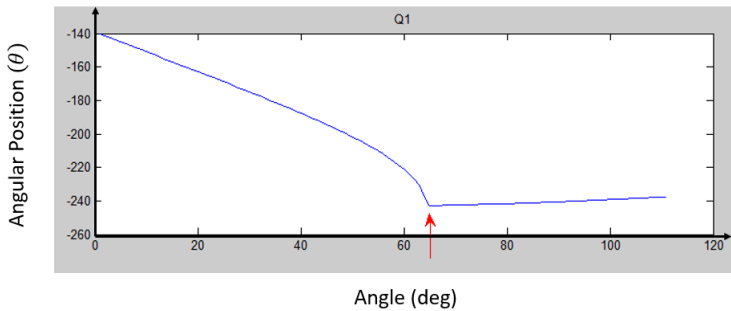
Figure 40 – Inverse singularity constraint for Y rotation, see red arrow.



Source: Own author.

which means it reached the physical limit.

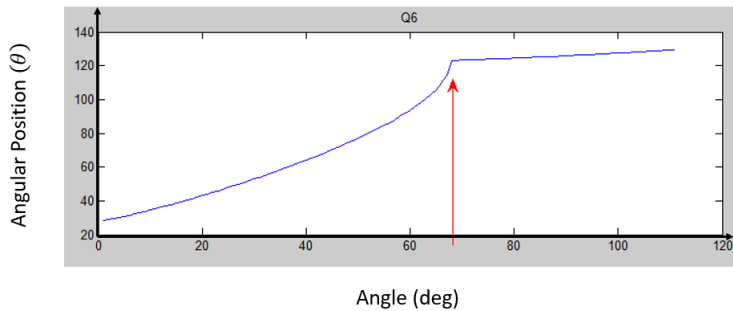
Figure 41 – Crank 1 movement for Y rotation, see red arrow.



Source: Own author.

6.5.2 Software Tool: ADAMS

This section presents CAE (Computed Aided Engineering) model analyses for IWF-robot and CEART-robot to confirm the developed algorithm results. Using the optimized parameters a CAE model for ADAMS is built to

Figure 42 – Crank 6 movement for Y rotation, see red arrow.

Source: Own author.

analyze 6-RUS optimized configuration. This parametric model is developed to enable the robot configuration update from a script. This script accesses another *.cmd* file containing the positions of the points A_i , B_i , C_i and the orientation of active joints to define 6-RUS in a new configuration. Then the optimized 6-RUS model is analyzed to verify the direct singularity and inverse singularity proximity.

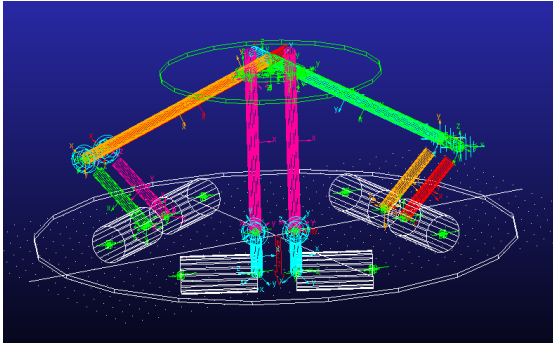
ADAMS (acronym of Automated Dynamic Analysis of Mechanical Systems) is a multibody dynamics simulation software equipped with Fortran and C++ numerical solvers. ADAMS was originally developed by Mechanical Dynamics Incorporation which then was acquired by MSC Software Corporation.

6.5.2.1 CAE Analyze: IWF-robot

Direct singularity problems mean that even with totally fixed limbs, there is an end-effector motion that may not be prevented. Therefore, in direct singular position, if forces/torques are applied to mobile end-effector, high

magnitude reaction force/torques may be appear in robot actuators. IWF-robot CAE layout may seen in Figure 43.

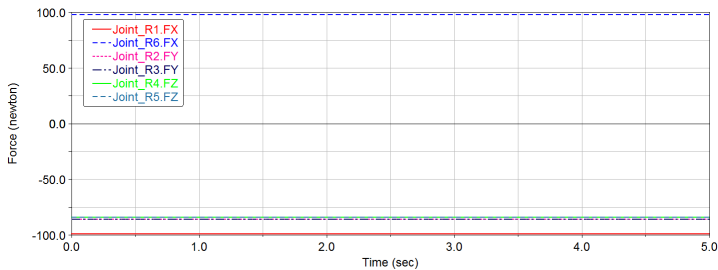
Figure 43 – IWF-robot layout in ADAMS.



Source: Own author.

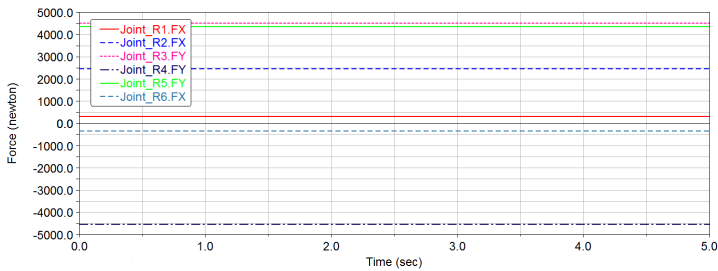
Constant torques in all axis are applied in IWF-robot end-effector in original position. The magnitude of the reaction forces is smaller them 100 Newtons (see Figure 44). The same constant torques are applied in IWF-robot end-effector into a singular position. As expected the reaction forces are significantly greater in magnitude, up to approximately 5000 Newtons (see Figure 45).

Figure 44 – IWF-robot torque application in initial position.



Source: Own author.

Figure 45 – IWF-robot torque application near direct singularity.

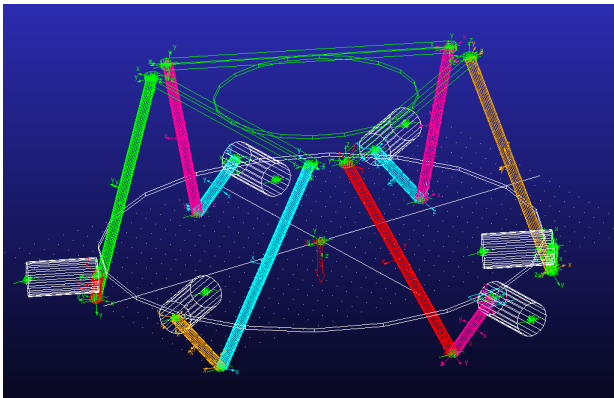


Source: Own author.

6.5.2.2 CAE Analyze: CEART-robot

CEART-robot have inverse singularity issues as Matlab demonstrates, see Section 6.3.2. Inverse singularity may be detected by ADAMS executing a movement until singular position. The CAE models movement should be the same as performed by program. CEART-robot CAE layout may seen in Figure 46.

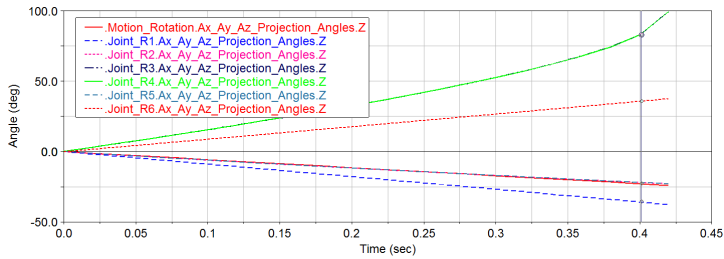
Figure 46 – CEART-robot layout in ADAMS.



Source: Own author.

Figure 47 shows that Cranks 3 and 4 have discontinuities in their movements at similar rotation angle 25° . Which means the legs are fully extended and ADAMS may automatic detect the inverse singularity when if some CAE model joint broken.

Figure 47 – CEART-robot rotation in Y axis, inverse singularity near 25° .



Source: Own author.

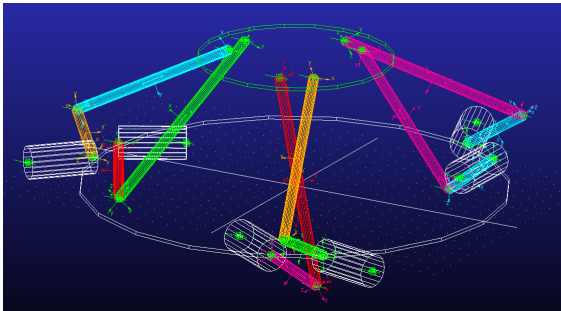
6.5.2.3 CAE Analyze: Optimized 6-RUS Robot

Optimized 6-RUS layout from ADAMS verified presents almost a symmetrical configuration (see Figure 48). As in Section 6.5.1, may verify the optimization results consistency, analyze the direct singularity proximity and the inverse singularity constraint, during Y rotation.

Constant torques are applied in Optimized 6-RUS end-effector in a singular position. As expected reaction forces reached high magnitudes, up to approximately 40000 Newtons, see (Figure 49).

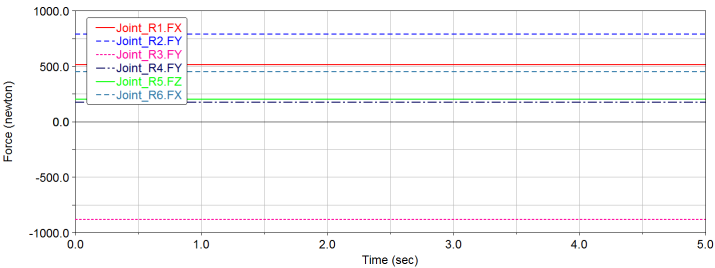
As Figure 50 show that Cranks 1 and 6 have discontinuities in there movement at similar rotation angle, about 65° , the robot reach inverse singular position. Therefore the CAE model demonstrated that Matlab algorithm have kinematic consistency.

Figure 48 – Optimized 6-RUS layout in ADAMS.



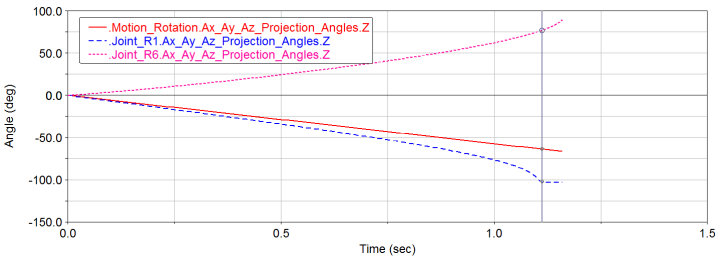
Source: Own author.

Figure 49 – Optimized 6-RUS robot torque application near direct singularity.



Source: Own author.

Figure 50 – Optimized 6-RUS movement for Y rotation.



Source: Own author.

7 CONCLUSIONS

This work presents a procedure to 6-RUS robot optimization to be used as flight simulator. This optimization method is based on direct/inverse singularity closeness and collision avoidance in order to maximize a 6-RUS parallel robot end-effector orientation, it is presented and verified using commercial CAE software. This method has direct application in flight simulators based on 6-RUS parallel robot where high orientation is an important project parameter. In order to obtain the optimum robot, some geometrical parameters are modified like, active joints location, the end-effector radius and end-effector height. The obtained robot is compared with IWF-robot and CEART-robot and a considerable advantage may be observed in the CAE software evaluation.

The model built by Ceart is taken as base, therefore the size should be similar. It is shown that IWF-robot also belongs to the same family of CEART-robot, but it has distinct performances, which motivate the optimization of 6-RUS configuration in order to work as flight simulation. The proposed task is maximize the rotation of the end-effector around all axes in XY plane. Therefore the orientation workspace should be optimized. The optimization is performed using the PSO based algorithm, in order to escape from local minima. The FMINCON function, configured to use the interior point algorithm is employed to accelerate the search for bottom of the valley indicated by the PSO as the best particles locations.

The optimized robot has better performance than both models mentioned above. Additionally, the CAE model in ADAMS demonstrated, through simulation, that the optimization method developed is consistent. It is observed

that symmetrical results are related in some way with crank and passive link radius, since these bodies may collide.

Just some parameters of 6-RUS kinematic chain are optimized, therefore a more complex optimization may be applied. The relationship between optimized non-symmetrical models and constraints may produce another analysis. Further studies about the kinematic relations and the forward kinematics index in this robot are interesting topics, as an extension of this work.

Some important points are the apparent between both relationship among forward kinematics index and the inverse kinematics constraint, and the mobile end-effector size. This relationship may be observed in three points: IWF robot configuration with little end-effector (compared with basis) have problem with direct singularity. In the Ceart robot bigger than IWF-robot mobile end-effector presents problems due to inverse singularity. Additionally, the optimized has a significantly larger end-effector than the IWF-robot. But smaller end-effector than CEART robot. However the range that show direct and inverse singularities are nearly the same.

This work motivates to build a real model with optimized configurations and develop studies also in other important areas such dynamics (rigid body movements). Other considerations on a prototype structure are the stiffness and the joints clearance that may also be considered in future works.

REFERENCE

- BALL, R. S. *A Treatise on the Theory of Screws*. Cambridge: Cambridge University Press, 1900. ISBN 0521636507 -reedicao 1998. Cited 2 times in pages 33 and 42.
- BARBOSA, M. R.; PIRES, E. S.; LOPES, A. M. Design optimization of a parallel manipulator based on evolutionary algorithms. v. 73, pp. p. 79?86, 2005. Cited in page 30.
- BONEV, I. A. *Geometric analysis of parallel mechanisms*. [S.l.: s.n.], 2003. 75–81 p. Cited in page 33.
- BROGARDH, T. Pkm research-important issues, as seen from a product development perspective at abb robotics. In: *Workshop on Fundamental Issues and Future Research Directions for Parallel Mechanisms and Manipulators, Quebec, Canada*. [S.l.: s.n.], 2002. Cited in page 29.
- CAMPOS, A. *Cálculo da Matriz Cinemática de Rede do Manipulador Paralelo 3RRR baseado na Lei de Kirchhoff e na Teoria de Helicóides*. Florianópolis, SC, 2001. Cited in page 33.
- CAMPOS, A. et al. Inverse kinematics for general 6-rus parallel robots applied on udesc-ceart flight simulator. In: *ABCM Symposium Series in Mechatronics*. [S.l.]: ABCM, Rio de Janeiro, RJ, Brazil, 2014. v. 6, p. PI.SIII.09,(456–464). Cited in page 97.
- CAMPOS, A.; GUENTHER, R.; MARTINS, D. Differential kinematics of serial manipulators using virtual chains. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, SciELO Brasil, v. 27, p. 345–356, 2005. Cited in page 63.
- CAMPOS, A. et al. Base angular layout optimization of the hexa parallel robot based on a singularity index. In: *21st International Congress of Mechanical Engineering, October, 24th-28th, Natal, Rio Grande do Norte, Brazil*. [S.l.: s.n.], 2011. Cited 4 times in pages 37, 38, 43, and 102.
- CLERC, M.; KENNEDY, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, IEEE, v. 6, n. 1, p. 58–73, 2002. Cited in page 86.
- DANDURAND, A. The rigidity of compound spatial grids. *Structural Topology*, n. 10, 1984. Cited in page 65.
- DANIALI, H. R. M.; ZSOMBOR-MURRAY, P. J.; ANGELES, J. Singularity analysis of planar parallel manipulators. *Mechanism and Machine Theory*, v. 30, n. 5, p. 665–678, 1995. Cited 2 times in pages 60 and 64.

DAVIDSON, J. K.; HUNT, K. H. *Robots and Screw Theory*. [S.l.]: Oxford University Press, 2004. Cited in page 58.

DAVIDSON, J. K.; HUNT, K. H. *Robots and Screw Theory: Applications of Kinematics and Statics to Robotics*. Nova Iorque: Oxford University Press Inc., 2004. ISBN 0-19-856245-4. Cited in page 62.

DONCKELS, B. *Particle Swarm Optimization, Shuffled Complex Evolution and SIMPSA (Nonlinear Simplex + Simulated Annealing)*. 2006. Disponível em: <<http://biomath.ugent.be/~brecht/index.html>>. Cited in page 85.

DUFFY, J. The fallacy of modern hybrid control theory that is based on 'orthogonal complements' of twist and wrench space. *Journal of Robotic Systems*, v. 7, n. 2, p. 139–144, 1990. Cited in page 68.

FICHTER, E. F. A stewart platform-based manipulator: general theory and practical construction. *The International Journal of Robotics Research*, Sage Publications, v. 5, n. 2, p. 157–182, 1986. Cited 2 times in pages 33 and 64.

FORSGREN, A.; GILL, P. E.; WRIGHT, M. H. Interior methods for nonlinear optimization. *SIAM review*, SIAM, v. 44, n. 4, p. 525–597, 2002. Cited in page 90.

FURTADO, C.; CAMPOS, A.; REIS, A. Orientation workspace optimization for a 6-rus parallel robot. In: *ABCM Symposium Series in Mechatronics*. [S.l.]: ABCM, Rio de Janeiro, RJ, Brazil, 2014. v. 6, p. PII.SIII.09,(1075–1084). Cited 3 times in pages 32, 36, and 97.

GOSSELIN, C. Kinematic analysis, optimization and programming of parallel robotic manipulators. *Journal of Mechanical Design*, American Society of Mechanical Engineers, v. 110, p. 35–41, 1988. Cited 3 times in pages 57, 59, and 60.

HAO, F.; MCCARTHY, J. Conditions for line-based singularities in spatial platform manipulators. *J. Rob. Syst.*, v. 15, n. 1, p. 43–55, 1998. Cited 2 times in pages 65 and 67.

HESSELBACH, J. et al. Direct kinematic singularity detection of a hexa parallel robot. Submitted to ICRA2005, IEEE. Cited in page 82.

HUNT, K. H. *Kinematic Geometry of Mechanisms*. Oxford: Clarendon Press, 1978. Cited in page 64.

HUNT, K. H. Structural kinematics of in-parallel-actuated robot-arms. *Trans. ASME, Journal of Mechanisms, Transmissions and Design*, v. 105, p. 705–712, Dec 1983. Cited 2 times in pages 33 and 34.

HUNT, K. H. *Kinematic geometry of mechanisms*. [S.l.]: Clarendon Press Oxford, 1990. Cited in page 67.

HUNT, K. H. Review: Don't cross-thread the screw!*. *Journal of Robotic Systems*, Wiley Online Library, v. 20, n. 7, p. 317–339, 2003. Cited 4 times in pages 38, 40, 41, and 42.

KACHITVICHYANUKUL, V. Comparison of three evolutionary algorithms: Ga, pso, and de. *Industrial Engineering & Management Systems*, v. 11, n. 3, p. 215–223, 2012. Cited in page 88.

KARMARKAR, N. A new polynomial-time algorithm for linear programming. In: ACM. *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. [S.l.], 1984. p. 302–311. Cited in page 90.

KELAIAIA, R.; ZAATRI, A. et al. Multiobjective optimization of a linear delta parallel robot. *Mechanism and Machine Theory*, Elsevier, v. 50, p. 159–178, 2012. Cited 2 times in pages 30 and 79.

KENNEDY, J. F.; KENNEDY, J.; EBERHART, R. C. *Swarm intelligence*. [S.l.]: Morgan Kaufmann, 2001. Cited 2 times in pages 85 and 86.

LAST, P. et al. Hexa-parallel-structure calibration by means of angular passive joint sensors. In: IEEE. *Mechatronics and Automation, 2005 IEEE International Conference*. [S.l.], 2005. v. 3, p. 1300–1305. Cited 3 times in pages 35, 82, and 98.

LIPKIN, H.; DUFFY, J. Hybrid twist and wrench control for a robotic manipulator. *Trans. ASME, Journal of Mechanisms, Transmissions and Design*, v. 110, p. 138–144, June 1988. Cited in page 68.

LIU, K. et al. The singularities and dynamics of a stewart platform manipulator. *Journal of Intelligent and Robotic Systems*, Springer, v. 8, n. 3, p. 287–308, 1993. Cited in page 64.

MEHROTRA, S. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, SIAM, v. 2, n. 4, p. 575–601, 1992. Cited in page 91.

MENEZES, A. J.; OORSCHOT, P. C. V.; VANSTONE, S. A. *Handbook of applied cryptography*. [S.l.]: CRC press, 1996. Cited in page 94.

MERLET, J. *Parallel Robots*. [S.l.]: Kluwer Academic Publisher, 2000. Cited 6 times in pages 34, 58, 62, 64, 65, and 66.

MERLET, J. P. Singular configurations of parallel manipulators and Grassman geometry. *International Journal of Robotics Research*, v. 8, n. 5, 1989. Cited 2 times in pages 65 and 67.

MERLET, J.-P. *Parallel robots*. [S.l.: s.n.], 2001. v. 74. Cited in page 33.

MERLET, J.-P. Jacobian, manipulability, condition number, and accuracy of parallel robots. *Journal of Mechanical Design*, American Society of Mechanical Engineers, v. 128, n. 1, p. 199–206, 2006. Cited in page 30.

MERLET, J.-P. *Parallel robots*. [S.l.]: Springer Science & Business Media, 2012. v. 74. Cited 2 times in pages 29 and 30.

MURRAY, R. M.; LI, Z.; SASTRY, S. S. *A Mathematical Introduction to Robotic Manipulation*. Ann Arbor: CRC Press, 1994. Cited in page 68.

PIERROT, F. A new design of a 6-dof parallel robot. *J. of Robotics and Mechatronics*, v. 2, n. 4, p. 308–315, 1990. Cited in page 34.

POINSOT, L. Sur la composition des moments et la composition des aires. *J. Éc Polyt. Paris*, v. 6, p. 182–205, 1806. Cited in page 40.

POTTMANN, H.; PETERNELL, M.; RAVANI, B. Approximation in line space—applications in robot kinematics and surface reconstruction. In: *Advances in robot kinematics: analysis and control (Salzburg, 1998)*. dordrecht: Kluwer Acad. Publ., 1998. p. 403–412. Cited 3 times in pages 69, 71, and 75.

RAO, S. S.; RAO, S. *Engineering optimization: theory and practice*. [S.l.]: John Wiley & Sons, 2009. Cited in page 85.

SCHUTTE, J. F.; GROENWOLD, A. A. A study of global optimization using particle swarms. *Journal of Global Optimization*, Springer, v. 31, n. 1, p. 93–108, 2005. Cited in page 89.

SCIAVICCO, L.; SICILIANO, B. *Modeling and Control of Robot Manipulators*. [S.l.]: McGraw-Hill, 1996. (Electrical and Computer Engineering). ISBN 0070572178. Cited 2 times in pages 47 and 59.

SICILIANO, B. et al. *Robotics: Modelling, Planning and Control*. Springer London, 2010. (Advanced Textbooks in Control and Signal Processing). ISBN 9781846286414. Disponível em: <<https://books.google.com.br/books?id=jPCAFmE-logC>>. Cited 2 times in pages 30 and 31.

STAN, S.-D.; MATIES, V.; BALAN, R. Multi-objective design optimisation of a planar micro parallel robot using genetic algorithms. *Journal of Control Engineering and Applied Informatics*, v. 9, n. 1, p. 41–46, 2007. Cited in page 30.

TAHMASEBI, F. *Kinematic Synthesis and Analysis of a Novel Class of Six-DOF Parallel Minimanipulators*. Tese (Doutorado) — The University of Maryland, 1992. Cited in page 64.

TSAI, L.-W. *Robot Analysis: the Mechanics of serial and parallel manipulators*. New York: John Wiley & Sons, 1999. ISBN 0-471-32593-7. Cited 6 times in pages 43, 57, 58, 59, 60, and 62.

VOGLEWEDE, P. *Measuring Closeness to Singularities of Parallel Manipulators with Application to the Design of Redundant Actuation*. Tese (Doutorado) — George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, 2004. Cited 7 times in pages 60, 67, 69, 71, 73, 74, and 76.

WANG, L.; ZHENG, D.-Z. An effective hybrid optimization strategy for job-shop scheduling problems. *Computers & Operations Research*, Elsevier, v. 28, n. 6, p. 585–596, 2001. Cited in page 95.

WOLF, A.; SHOHAM, M. Investigation of parallel manipulators using linear complex approximation. *ASME, Journal of Mechanical Design*, v. 125, n. 3, p. 564–572, 2003. Cited 2 times in pages 70 and 71.

WOLF, A.; SHOHAM, M. Investigation of parallel manipulators using linear complex approximation. *Journal of Mechanical Design*, American Society of Mechanical Engineers, v. 125, n. 3, p. 564–572, 2003. Cited in page 75.

XU, Y. X.; KOHLI, D.; WENG, T. C. Direct differential kinematics of hybrid-chain manipulators including singularity and stability analyses. *Journal of Mechanical Design*, v. 116, n. 2, p. 614–621, jun. 1994. Cited in page 64.

YOSHIKAWA, T. *Foundations of Robotics: Analysis and Control*. [S.l.]: MIT Press, 1990. Cited in page 68.

ZABALZA, I. et al. Total and partial stationary configurations for a 6-rus hunt-type parallel manipulator. In: *Proceedings of the 11th world congress in mechanism and machine science, IFTOMM*. [S.l.: s.n.], 2003. p. 18–21. Cited in page 34.

ZHANG, D. Global stiffness optimization of parallel robots using kinetostatic performance indices. 2010. Cited in page 30.

8 APPENDIX A: MATLAB ALGORITHM

The Appendix A presents the kinematical and screw based algorithm developed in Matlab by this work and also the optimization method with all sub programs.

8.1 Main Program

```
clear all

%----- MAIN -----
%----- PROGRAM -----
%---- INDEX Based in Work -----

global limit_cine_dire;

% -----
%---- INDEX Based in Work Limit Value -----

limit_cine_dire=0.0293;

% -----
% ----- CONSTRAINTS LIMITS -----

global limit_cine_inv;
```

```

global limit_dist_limb;
global limit_crank;

% -----

% ---- CONSTRAINS LIMITS  VALUES -----

limit_dist_limb=0.03;  % limbs collision
limit_crank=0.031;    % Cranks collision
limit_cine_inv=1;      % Inverse Kinematics

%-----

%----- START -----
%----- OPTIMIZATION -----

% ----- Inicial Values -----
%X0=[Beta1 Beta2 Beta3 Beta4 Beta5 Beta6 rP e f Z];
X0=[66 160 180 284 302 44 0.1972  0.0766  0.0501  0.4042];

% -----Superior Limit -----
Lb=[0,0,0,0,0,0,0.1,0.01,0.01,0.2];

% -----Inferior Limit -----
Ub=[360,360,360,360,360,360,0.7,0.7,0.7,0.8];

% -----

```

```

%----- START -----
%----- PSO -----

%----- PSO Configuration -----

options = PSOSET('SWARM_SIZE', 30 , 'MAX_ITER', 3000,
'COGNITIVE_ACC', 2.8, 'SOCIAL_ACC', 1.3);

%-----PSO-----

[X1 , Fval] = PSO('Prog_1',X0,Lb,Ub,options);

%----- END PSO -----
%-----

e_inicial=X1;

%-----
%----- START -----
%----- FMINCON -----
%-----
%----- FMINCON Configuration ---
%-----

options2 = optimset('Algorithm','interior-point',

```

```

'Display','iter','GradObj','on','TolFun',1.5,
'AlwaysHonorConstraints','none','Display',
'iter','PlotFcns',{@optimplotx, @optimplotfval,
    @optimplotconstrviolation,
    @optimplotfirstorderopt,@optimplotstepsize},
'InitBarrierParam',0.05);

%-----FMINCON -----

[X2 xfval exitflag output] =fmincon(@Prog_2,
e_inicial,[],[],[],[],[0 0 0 0 0 0 0.1 0.1
    0.02 0.1], [360 360 360 360 360 360 1 1 1 1],
@Constrain_grad,options2);

%-----
%-----      END FMINCON -----
%-----

%-----
%-----      END OPTIMIZATION -----
%-----

```

8.2 Program 1

```
function Y = Prog_1(X)

% Return Program 10 Result in Y

    Y=Kinematic_Program(10,X);

end
```


8.3 Kinematic and Screw Based Program

```

function W = Kinematic_Program(prog,X)

rad = pi/180;

%---- INDEX Based in Work -----

global limit_cine_dire;

% ----- CONSTRAINS LIMITS -----

global limit_cine_inv;
global limit_dist_limb;
global limit_crank;
% -----

for alpha=0:1:360
    delta=(alpha)*rad;
    Fun_Obja=50;
    FDE_R=5;

for kappa=0:1:120
    hhh=hhh+1;

%-----
%Given Position of the TCP in [m] and [degrees]

```

```

%-----

TCP=[0 0 -X(10) (kappa*cos(delta)) (kappa*sin(delta)) 0]';

%-----

% TCP angles from degrees to radians
%-----

TCP(4) = TCP(4) *rad;
TCP(5) = TCP(5) *rad;
TCP(6) = TCP(6) *rad;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CEART'S FLIGHT SIMULATOR DATA %%%%%%%%%%

rB = 0.636885;% radius of the Base [m] % (rB)
li = 0.194; % length of the first link [m] % (li)
Li = 0.650; % length of the passive link [m] % (Li)

xsi(1:6) = [0 120 120 240 240 0]*rad;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% OPTIMIZATION PARAMETERS %%%%%%%%%%

rP = X(7);%radius of the moving plattform[m]%(rP)
e = X(8);%half distance between actuators[m]%(e)

```

```

f    = X(9);%half distance between joints[m]%(f)

zeta(1:6) = [X(1) X(2) X(3) X(4) X(5) X(6)]*rad;

PTCP= [0 0 0]';

cXsi = cos(xsi);
sXsi = sin(xsi);

cZeta= cos(zeta);
sZeta= sin(zeta);

%-----
% Computation of the plattform vectors from the middle
% -----to the actuators/joints-----
%-----

for i=1:6
    m = (-1)^(i-1)*e;
    n = (-1)^(i-1)*f;
    A(:,i) = [rB*cos(xsi(i))-m*sin(xsi(i)); ...
rB*sin(xsi(i))+m*cos(xsi(i)); 0];

```

```

    P(:,i) = [rP*cos(xsi(i))-n*sin(xsi(i)); ...
    rP*sin(xsi(i))+n*cos(xsi(i)); 0] - P_TCP;

end

%-----

cosPsi    = cos(TCP(4));
sinPsi    = sin(TCP(4));
cosTheta  = cos(TCP(5));
sinTheta  = sin(TCP(5));
cosPhi    = cos(TCP(6));
sinPhi    = sin(TCP(6));

Rot=[cosPhi*cosTheta cosPhi*sinTheta*sinPsi-sinPhi*cosPsi
    cosPhi*sinTheta*cosPsi+sinPhi*sinPsi;
    sinPhi*cosTheta sinPhi*sinTheta*sinPsi+cosPhi*cosPsi
    sinPhi*sinTheta*cosPsi-cosPhi*sinPsi;
    -sinTheta cosTheta*sinPsi cosTheta*cosPsi];

C = Rot * P;

% -----
% ---- Begin IKP HexaII -----

```

```

% -----

for i=1:6

    IKP_g = A(:,i) - C(:,i) - TCP(1:3);

    IKP_u = 2*li*(IKP_g(1)*cZeta(i) + IKP_g(2)*sZeta(i));

    IKP_v = -2 * li * IKP_g(3);

    IKP_w = li^2-li^2-IKP_g(1)^2-IKP_g(2)^2-IKP_g(3)^2;

    IKP_N = real(sqrt(IKP_u^2 + IKP_v^2 - IKP_w^2));

    aaa=IKP_u^2 + IKP_v^2;
    bbb=IKP_w^2;
    ccc(i,1)=aaa/bbb;

%-----

q(i,1) = pi - atan2(IKP_w, IKP_N) - atan2(IKP_u, IKP_v);
if (i==1 | i==3 | i==5)
q(i,1) = pi/2 - atan2(IKP_N,IKP_w) - atan2(IKP_u,IKP_v);
end

%-----

```

```

end

% -----
% ----- End IKP RUS -----
% -----

% -----
% ----- Begin Jacobian Matrix -----

toleranz = 1e-6;

for i=1:6
    b(:,i) = [cos(zeta(i))*cos(q(i)); sin(zeta(i))*cos(q(i));
    -sin(q(i))]*li;
end

B = b + A;

TCPMatlix = repmat(TCP(1:3),1,6);

c = -TCPMatlix - C + B;

%-----
% Differentation of the Kardan angles with Sciavicco
%-----

DkardanDPsi = [0
    cosPhi*sinTheta*cosPsi+sinPhi*sinPsi

```

```

-cosPhi*sinTheta*sinPsi+sinPhi*cosPsi;
0
sinPhi*sinTheta*cosPsi-cosPhi*sinPsi
-sinPhi*sinTheta*sinPsi-cosPhi*cosPsi;
0
cosTheta*cosPsi -cosTheta*sinPsi];

```

```

DkardanDTheta =[-cosPhi*sinTheta cosPhi*cosTheta*sinPsi
cosPhi*cosTheta*cosPsi; -sinPhi*sinTheta
sinPhi*cosTheta*sinPsi sinPhi*cosTheta*cosPsi;
-cosTheta -sinTheta*sinPsi -sinTheta*cosPsi];

```

```

DkardanDPhi = [-sinPhi*cosTheta
-sinPhi*sinTheta*sinPsi-cosPhi*cosPsi
-sinPhi*sinTheta*cosPsi+cosPhi*sinPsi;
cosPhi*cosTheta cosPhi*sinTheta*sinPsi-sinPhi*cosPsi
cosPhi*sinTheta*cosPsi+sinPhi*sinPsi;
0 0 0];

```

```

JacobiMatlix = -2 * c';

```

```

for i=1:6

```

```

JacobiMatlix(i,4)=JacobiMatlix(i,1:3)*DkardanDPsi*P(:,i);

```

```

JacobiMatlix(i,5)=JacobiMatlix(i,1:3)*DkardanDTheta*P(:,i);

```

```

JacobiMatlix(i,6)=JacobiMatlix(i,1:3)*DkardanDPhi*P(:,i);

```

```

end

% -----
% ---- End Jacobian Matrix -----
% -----

%-----
%Vector from Base to the Point Ci
%-----

CiWKS = C + TCPMatlix;

%-----
%-----
%Vector from point Bi to Ci
%-----

Bi_Ci_WKS = CiWKS - B;

%-----

%the given lines which are the wrenches in axis-order
%-----

n=norm(Bi_Ci_WKS(:,1));

L_1=[cross(CiWKS(:,1),Bi_Ci_WKS(:,1));...
```



```

Bi_Ci_WKS(:,1)]/n;
L_2=[cross(CiWKS(:,2),Bi_Ci_WKS(:,2));...
Bi_Ci_WKS(:,2)]/n;
L_3=[cross(CiWKS(:,3),Bi_Ci_WKS(:,3));...
Bi_Ci_WKS(:,3)]/n;
L_4=[cross(CiWKS(:,4),Bi_Ci_WKS(:,4));...
Bi_Ci_WKS(:,4)]/n;
L_5=[cross(CiWKS(:,5),Bi_Ci_WKS(:,5));...
Bi_Ci_WKS(:,5)]/n;
L_6=[cross(CiWKS(:,6),Bi_Ci_WKS(:,6));...
Bi_Ci_WKS(:,6)]/n;

%-----

%The Gramian matrix which represents the set of lines
%-----

M=L_1*L_1'+L_2*L_2'+L_3*L_3'+L_4*L_4'+L_5*L_5'+L_6*L_6';

%-----

%-----

%Matrix with the diagonal elements:
%-----

D = eye(3);
D(6,6) = 0;

```

```

%-----
%The determinant has to become zero
%-----

de=det (M) ;

%-----
%MMM defines the eigenvalue problem
%-----

MMM=inv (M) *D;

%-----

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Solving the eigenvalue problem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[evect,eval]=eig (MMM) ;

desvi_std=sqrt (eval (4,4)+eval (5,5)+eval (6,6)) ;

%-----
%seperate the maximum eigenvalue and his eigenvector
%-----

```

```

[max_auto, pos_max_auto]=max(max(eval));

%-----
%The distance of the eigenvector to the set of lines
%
%           is the work
%-----

lam=1/max_auto;

%-----
%The following equations are needed to draw the axis
%of the founded screw the relationship between the angle
%velocity and the velocity around the screw axis defines
%the pitch of the screw
%-----

Complx=evect(:,pos_max_auto);

%-----
%Here are the components of the screw
%-----

dire_C=[Complx(1) Complx(2) Complx(3)];
mome_C=[Complx(4) Complx(5) Complx(6)];

```

%----limbs and Cranks Colission -----

```
A_A = [1, A(1,1),A(2,1),A(3,1);...
        2, A(1,2),A(2,2),A(3,2);...
        3, A(1,3),A(2,3),A(3,3);...
        4, A(1,4),A(2,4),A(3,4);...
        5, A(1,5),A(2,5),A(3,5);...
        6, A(1,6),A(2,6),A(3,6)];
```

```
A_B = [1, B(1,1),B(2,1),B(3,1);...
        2, B(1,2),B(2,2),B(3,2);...
        3, B(1,3),B(2,3),B(3,3);...
        4, B(1,4),B(2,4),B(3,4);...
        5, B(1,5),B(2,5),B(3,5);...
        6, B(1,6),B(2,6),B(3,6)];
```

```
A_C = [1, CiWKS(1,1),CiWKS(2,1),CiWKS(3,1);...
        2, CiWKS(1,2),CiWKS(2,2),CiWKS(3,2);...
        3, CiWKS(1,3),CiWKS(2,3),CiWKS(3,3);...
        4, CiWKS(1,4),CiWKS(2,4),CiWKS(3,4);...
        5, CiWKS(1,5),CiWKS(2,5),CiWKS(3,5);...
        6, CiWKS(1,6),CiWKS(2,6),CiWKS(3,6)];
```

% ----- limb 1 and 6 -----

```

V_1=CiWKS(:,1)-B(:,1);
V_6=CiWKS(:,6)-B(:,6);

a1_6=B(:,6)-B(:,1);

N1_6=cross(V_1,V_6);
N_uni1_6=(N1_6)/norm(N1_6);

Dist1_6=norm(a1_6'*N_uni1_6);%Distancia entre limbs

% Cacula pontos
[A1e6,B1e6,C1e6,D1e6]=
Calcula_pontos_ABCD_1e6(li,e,zeta(1),zeta(6));

% crank 1 e 6
[distance1e6 varargout1e6]=
DistBetween2Segment(A1e6, B1e6, C1e6, D1e6);

%-----
% ----- limb 2 and 3 -----
V_2=CiWKS(:,2)-B(:,2);
V_3=CiWKS(:,3)-B(:,3);

a2_3=B(:,3)-B(:,2);

```

```

N2_3=cross(V_2,V_3);
N_uni2_3=(N2_3)/norm(N2_3);

Dist2_3=norm(a2_3'*N_uni2_3); % Distancia entre limbs

% Cacula pontos
[A2e3,B2e3,C2e3,D2e3]=
Calcula_pontos_ABCD_2e3(li,e,zeta(2),zeta(3));

% crank 2 e 3
[distance2e3 varargout2e3]=
DistBetween2Segment(A2e3, B2e3, C2e3, D2e3);
%-----
% ----- limb 4 and 5 -----
V_4=CiWKS(:,4)-B(:,4);
V_5=CiWKS(:,5)-B(:,5);

a4_5=B(:,5)-B(:,4);

N4_5=cross(V_4,V_5);
N_uni4_5=(N4_5)/norm(N4_5);

Dist4_5=norm(a4_5'*N_uni4_5); % Distancia entre limbs

% Cacula pontos

```

[illegible]

```

EPS=[distance1e6 distance2e3 distance4e5];

eps=min(EPS);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ----- CONSTRAINS LIMITS -----

    if (dist<limit_dist_limb)
        Fun_Obja=hh;
        FDE_R=1;
        break;
    end

    if (eps<limit_crank)
        Fun_Obja=hh;
        FDE_R=2;
        break;
    end

    if (ICI<limit_cine_inv)
        Fun_Obja=hh;
        FDE_R=3;
        break;
    end

```



```

%---- INDEX Based in Work Limit Value -----
    if (work<limit_cine_dire)
        Fun_Obja=hh;
        FDE_R=4;
        break;
    end

end

Work(alpha+1)=work;
VICI(alpha+1)=ICI;
DDist(alpha+1)=dist;
EEps(alpha+1)=eps;
Fun_Objb(alpha+1)=Fun_Obja;

FER(alpha+1)=FDE_R;

end

[Fun_ObjP,I]=min(Fun_Objb);

IWork=Work(I);
IICI=VICI(I);
Idist=DDist(I);
Ieps=EEps(I);

```

```
IFDER=FER(I)
```

```
if (prog==5)
```

```
W=Idist;
```

```
if (prog==6)
```

```
W=Ieps;
```

```
if (prog==7)
```

```
W=IWork;
```

```
if (prog==8)
```

```
W=IICI;
```

```
elseif (prog==10)
```

```
W=-Fun_ObjP;
```

```
end
```

8.4 Program 2

```
function [F,G] = Prog_2(e)
```

```
F=Prog_1(e);
```

```
% Return Gradient of Function Objective
```

```
G=F_Diff_Function_Obj(e);
```

```
end
```

8.5 Function Objective Gradient

```
function dwde = F_Diff_Function_Obj(e)

delta_e=[1 1 1 1 1 1 5e-2 5e-2 5e-2 5e-2];

lam_e_mais_e_1=Prog_1(e+[delta_e(1) 0 0 0 0
0 0 0 0 0]);
lam_e_menos_e_1=Prog_1(e-[delta_e(1) 0 0 0 0
0 0 0 0 0]);

lam_e_mais_e_2=Prog_1(e+[0 delta_e(2) 0 0 0
0 0 0 0 0]);
lam_e_menos_e_2=Prog_1(e-[0 delta_e(2) 0 0 0
0 0 0 0 0]);

lam_e_mais_e_3=Prog_1(e+[0 0 delta_e(3) 0 0
0 0 0 0 0]);
lam_e_menos_e_3=Prog_1(e-[0 0 delta_e(3) 0 0
0 0 0 0 0]);

lam_e_mais_e_4=Prog_1(e+[0 0 0 delta_e(4) 0
```

```

0 0 0 0 0]);
lam_e_menos_e_4=Prog_1(e-[0 0 0 delta_e(4) 0
0 0 0 0 0]);

lam_e_mais_e_5=Prog_1(e+[0 0 0 0 delta_e(5)
0 0 0 0 0]);
lam_e_menos_e_5=Prog_1(e-[0 0 0 0 delta_e(5)
0 0 0 0 0]);

lam_e_mais_e_6=Prog_1(e+[0 0 0 0 0
delta_e(6) 0 0 0 0]);
lam_e_menos_e_6=Prog_1(e-[0 0 0 0 0
delta_e(6) 0 0 0 0]);

lam_e_mais_e_7=Prog_1(e+[0 0 0 0 0 0
delta_e(7) 0 0 0]);
lam_e_menos_e_7=Prog_1(e-[0 0 0 0 0 0
delta_e(7) 0 0 0]);

lam_e_mais_e_8=Prog_1(e+[0 0 0 0 0 0 0
delta_e(8) 0 0]);
lam_e_menos_e_8=Prog_1(e-[0 0 0 0 0 0 0
delta_e(8) 0 0]);

lam_e_mais_e_9=Prog_1(e+[0 0 0 0 0 0 0

```

```

0 delta_e(9) 0]);
lam_e_menos_e_9=Prog_1(e-[0 0 0 0 0 0 0
0 delta_e(9) 0]);

lam_e_mais_e_10=Prog_1(e+[0 0 0 0 0 0 0
0 0 delta_e(10)]);
lam_e_menos_e_10=Prog_1(e-[0 0 0 0 0 0 0
0 0 delta_e(10)]);

dwde_1=(lam_e_mais_e_1-lam_e_menos_e_1)
/(2*delta_e(1));
dwde_2=(lam_e_mais_e_2-lam_e_menos_e_2)
/(2*delta_e(2));
dwde_3=(lam_e_mais_e_3-lam_e_menos_e_3)
/(2*delta_e(3));
dwde_4=(lam_e_mais_e_4-lam_e_menos_e_4)
/(2*delta_e(4));
dwde_5=(lam_e_mais_e_5-lam_e_menos_e_5)
/(2*delta_e(5));
dwde_6=(lam_e_mais_e_6-lam_e_menos_e_6)
/(2*delta_e(6));
dwde_7=(lam_e_mais_e_7-lam_e_menos_e_7)
/(2*delta_e(7));
dwde_8=(lam_e_mais_e_8-lam_e_menos_e_8)

```

```
/(2*delta_e(8));  
dwde_9=(lam_e_mais_e_9-lam_e_menos_e_9)  
/(2*delta_e(9));  
dwde_10=(lam_e_mais_e_10-lam_e_menos_e_10)  
/(2*delta_e(10));  
  
dwde=[dwde_1 dwde_2 dwde_3 dwde_4 dwde_5 dwde_6 dwde_7  
dwde_8 dwde_9 dwde_10];
```

8.6 Constraints Program

```

function [c ceq gradc gradceq] = Constrain_grad(X)

global limit_dist_limb;
global limit_crank;
global limit_cine_inv;
global limit_cine_dire;

c=[];

gradc=[];

% ----- CONSTRAINS LIMITS -----

c(1)=limit_dist_limb-p4_novo(5,X);
c(2)=limit_crank-p4_novo(6,X);
c(3)=limit_cine_dire-p4_novo(7,X);
c(4)=limit_cine_inv-p4_novo(8,X);

% --- Gradient Matrix for Each Optimization Parameters

```


and Constrains-----

```

gradc=[-F_Diff(5,1,X), -F_Diff(6,1,X), -F_Diff(7,1,X),
        -F_Diff(8,1,X);...
        -F_Diff(5,2,X), -F_Diff(6,2,X), -F_Diff(7,2,X),
        -F_Diff(8,2,X);...
        -F_Diff(5,3,X), -F_Diff(6,3,X), -F_Diff(7,3,X),
        -F_Diff(8,3,X);...
        -F_Diff(5,4,X), -F_Diff(6,4,X), -F_Diff(7,4,X),
        -F_Diff(8,4,X);...
        -F_Diff(5,5,X), -F_Diff(6,5,X), -F_Diff(7,5,X),
        -F_Diff(8,5,X);...
        -F_Diff(5,6,X), -F_Diff(6,6,X), -F_Diff(7,6,X),
        -F_Diff(8,6,X);...
        -F_Diff(5,7,X), -F_Diff(6,7,X), -F_Diff(7,7,X),
        -F_Diff(8,7,X);...
        -F_Diff(5,8,X), -F_Diff(6,8,X), -F_Diff(7,8,X),
        -F_Diff(8,8,X);...
        -F_Diff(5,9,X), -F_Diff(6,9,X), -F_Diff(7,9,X),
        -F_Diff(8,9,X);...
        -F_Diff(5,10,X), -F_Diff(6,10,X), -F_Diff(7,10,X),
        -F_Diff(8,10,X)];

```

```
ceq = [];
```

```
gradceq = [];
```

8.7 Constraints Finite Differences Program

Constrains Program

```
function dwde = F_Diff_Constrains(p,v,e)

delta_e=[1 1 1 1 1 1 5e-2 5e-2 5e-2 5e-2];

if(v==1)
lam_e_mais_e_1=Grad_Constrains(p,e+[delta_e(1)
0 0 0 0 0 0 0 0 0]);
lam_e_menos_e_1=Grad_Constrains(p,e-[delta_e(1)
0 0 0 0 0 0 0 0 0]);

dwde_1=(lam_e_mais_e_1-lam_e_menos_e_1)/(2*delta_e(1));

dwde=[dwde_1 0 0 0 0 0 0 0 0 0];

elseif(v==2)
lam_e_mais_e_2=Grad_Constrains(p,e+[0 delta_e(2)
0 0 0 0 0 0 0 0]);
lam_e_menos_e_2=Grad_Constrains(p,e-[0 delta_e(2)
```

```

0 0 0 0 0 0 0 0]);

dwde_2=(lam_e_mais_e_2-lam_e_menos_e_2)/(2*delta_e(2));

dwde=[0 dwde_2 0 0 0 0 0 0 0];

elseif(v==3)
lam_e_mais_e_3=Grad_Contrains(p,e+[0 0 delta_e(3)
0 0 0 0 0 0 0]);
lam_e_menos_e_3=Grad_Contrains(p,e-[0 0 delta_e(3)
0 0 0 0 0 0 0]);

dwde_3=(lam_e_mais_e_3-lam_e_menos_e_3)/(2*delta_e(3));

dwde=[0 0 dwde_3 0 0 0 0 0 0];

elseif(v==4)
lam_e_mais_e_4=Grad_Contrains(p,e+[0 0 0 delta_e(4)
0 0 0 0 0 0]);
lam_e_menos_e_4=Grad_Contrains(p,e-[0 0 0 delta_e(4)
0 0 0 0 0 0]);

dwde_4=(lam_e_mais_e_4-lam_e_menos_e_4)/(2*delta_e(4));

dwde=[0 0 0 dwde_4 0 0 0 0 0];

```

```
elseif(v==5)

lam_e_mais_e_5=Grad_Contrains(p,e+[0 0 0 0 delta_e(5)
0 0 0 0 0]);

lam_e_menos_e_5=Grad_Contrains(p,e-[0 0 0 0 delta_e(5)
0 0 0 0 0]);


dwde_5=(lam_e_mais_e_5-lam_e_menos_e_5)/(2*delta_e(5));


dwde=[0 0 0 0 dwde_5 0 0 0 0 0];


elseif(v==6)

lam_e_mais_e_6=Grad_Contrains(p,e+[0 0 0 0 0 delta_e(6)
0 0 0 0]);

lam_e_menos_e_6=Grad_Contrains(p,e-[0 0 0 0 0 delta_e(6)
0 0 0 0]);


dwde_6=(lam_e_mais_e_6-lam_e_menos_e_6)/(2*delta_e(6));


dwde=[0 0 0 0 0 dwde_6 0 0 0 0];


elseif(v==7)

lam_e_mais_e_7=Grad_Contrains(p,e+[0 0 0 0 0 0
delta_e(7) 0 0 0]);
```

```

lam_e_menos_e_7=Grad_Contrains(p,e-[0 0 0 0 0 0
delta_e(7) 0 0 0]);

dwde_7=(lam_e_mais_e_7-lam_e_menos_e_7)/(2*delta_e(7));

dwde=[0 0 0 0 0 0 dwde_7 0 0 0];

elseif(v==8)
lam_e_mais_e_8=Grad_Contrains(p,e+[0 0 0 0 0 0 0
delta_e(8) 0 0]);
lam_e_menos_e_8=Grad_Contrains(p,e-[0 0 0 0 0 0 0
delta_e(8) 0 0]);

dwde_8=(lam_e_mais_e_8-lam_e_menos_e_8)/(2*delta_e(8));

dwde=[0 0 0 0 0 0 0 dwde_8 0 0];

elseif(v==9)
lam_e_mais_e_9=Grad_Contrains(p,e+[0 0 0 0 0
0 0 0 delta_e(9) 0]);
lam_e_menos_e_9=Grad_Contrains(p,e-[0 0 0 0 0
0 0 0 delta_e(9) 0]);

dwde_9=(lam_e_mais_e_9-lam_e_menos_e_9)/(2*delta_e(9));

```

```
dwde=[0 0 0 0 0 0 0 0 dwde_9 0];

elseif(v==10)
lam_e_mais_e_10=Grad_Contrains(p,e+[0 0 0 0 0 0
0 0 0 delta_e(10)]);
lam_e_menos_e_10=Grad_Contrains(p,e-[0 0 0 0 0 0
0 0 0 delta_e(10)]);

dwde_10=(lam_e_mais_e_10-lam_e_menos_e_10)/(2*delta_e(10));

dwde=[0 0 0 0 0 0 0 0 dwde_10];

end
```

8.8 Constraints Gradient Program

```
function W = Grad_Contrains(prog,X)
```

```
W=Kinematic_Program(prog,X)
```

```
end
```


9 APPENDIX B: ADAMS SCRIPT

The Appendix B presents ADAMS parametric mode script in *.cmd* and another *.cmd* file containing the positions points A_i , B_i , C_i and the orientation of active joints to define 6-RUS in a new configuration.

9.1 ADAMS Parametric Model Script

In this Appendix are appended algorithms

```
!
!----- Default Units for Model -----!
!
!
defaults units &
    length = meter &
    angle = deg &
    force = newton &
    mass = kg &
    time = sec
!
defaults units &
    coordinate_system_type = cartesian &
    orientation_type = body313
!
```

```
!----- Default Attributes for Model -----!  
!  
!  
defaults attributes &  
    inheritance = bottom_up &  
    icon_visibility = on &  
    grid_visibility = off &  
    size_of_icons = 5.0E-002 &  
    spacing_for_grid = 1.0  
!  
!----- Adams/View Model -----!  
!  
!  
model create &  
    model_name = MODEL_1  
!  
view erase  
!  
!----- Materials ---!  
!  
!  
material create &  
    material_name = .MODEL_1.steel &  
    adams_id = 1 &  
    youngs_modulus = 2.07E+011 &
```

```
    poissos_ratio = 0.29  &
    density = 7801.0
!
!----- Rigid Parts ---!
!
! Create parts and their dependent markers and graphics
!
!----- ground ---!
!
!
! ***** Ground Part *****
!
defaults model  &
    part_name = ground
!
defaults coordinate_system  &
    default_coordinate_system = .MODEL_1.ground
!
! ***** Markers for current part *****
!
marker create  &
    marker_name = .MODEL_1.ground.MARKER_63  &
    adams_id = 63  &
    location = 0.0, 0.0, -0.45  &
    orientation = 180.0d, 90.0d, 180.0d
```

!

```
marker create &  
    marker_name = .MODEL_1.ground.MARKER_1 &  
    adams_id = 1 &  
    location = 0.0, 0.0, 0.0 &  
    orientation = 0.0d, 0.0d, 0.0d
```

!

```
marker create &  
    marker_name = .MODEL_1.ground.MARKER_3 &  
    adams_id = 3 &  
    location = -0.14735, 0.65034, 0.0 &  
    orientation = 180.0d, 90.0d, 180.0d
```

!

```
marker create &  
    marker_name = .MODEL_1.ground.MARKER_4 &  
    adams_id = 4 &  
    location = -0.48953, 0.45278, 0.0 &  
    orientation = 240.0007046147d, 90.0d, 90.0d
```

!

```
marker create &  
    marker_name = .MODEL_1.ground.MARKER_5 &  
    adams_id = 5 &  
    location = 0.63689, 0.19756, 0.0 &  
    orientation = 120.0007277808d, 90.0d, 270.0d
```

!

```
marker create &

    marker_name = .MODEL_1.ground.MARKER_6 &
    adams_id = 6 &
    location = 0.63689, -0.19756, 0.0 &
    orientation = 59.9992722192d, 90.0d, 270.0d
!
```

```
marker create &

    marker_name = .MODEL_1.ground.MARKER_7 &
    adams_id = 7 &
    location = -0.14735, -0.65034, 0.0 &
    orientation = 0.0d, 90.0d, 0.0d
!
```

```
marker create &

    marker_name = .MODEL_1.ground.MARKER_8 &
    adams_id = 8 &
    location = -0.48953, -0.45278, 0.0 &
    orientation = 299.9992953853d, 90.0d, 90.0d
!
```

```
marker create &

    marker_name = .MODEL_1.ground.MARKER_28 &
    adams_id = 28 &
    location = -0.48953, 0.45278, 0.0 &
    orientation = 240.0007046147d, 90.0d, 90.0d
!
```

```
marker create &
```

```
marker_name = .MODEL_1.ground.MARKER_30  &
adams_id = 30  &
location = -0.14735, 0.65034, 0.0  &
orientation = 180.0d, 90.0d, 180.0d
!

marker create  &
marker_name = .MODEL_1.ground.MARKER_32  &
adams_id = 32  &
location = 0.63689, 0.19756, 0.0  &
orientation = 120.0007277808d, 90.0d, 270.0d
!

marker create  &
marker_name = .MODEL_1.ground.MARKER_78  &
adams_id = 78  &
location = 0.63689, -0.19756, 0.0  &
orientation = 59.9992722192d, 90.0d, 270.0d
!

marker create  &
marker_name = .MODEL_1.ground.MARKER_36  &
adams_id = 36  &
location = -0.14735, -0.65034, 0.0  &
orientation = 0.0d, 90.0d, 0.0d
!

marker create  &
marker_name = .MODEL_1.ground.MARKER_38  &
```

```
    adams_id = 38    &
    location = -0.48953, -0.45278, 0.0    &
    orientation = 299.9992953853d, 90.0d, 90.0d
!
part create rigid_body mass_properties    &
    part_name = .MODEL_1.ground    &
    material_type = .MODEL_1.steel
!
! ***** Points for current part *****
!
point create    &
    point_name = .MODEL_1.ground.Origem    &
    location = 0.0, 0.0, 0.0
!
!
!
file command read file_name="H:\Adams\6RUS_Otimi.cmd"
!
!
! ***** Graphics for current part *****
!
geometry create shape cylinder    &
    cylinder_name = .MODEL_1.ground.Base    &
    adams_id = 2    &
    center_marker = .MODEL_1.ground.MARKER_1    &
```



```
angle_extent = 360.0  &
length = 1.0E-002  &
radius = 0.636885  &
side_count_for_body = 20  &
segment_count_for_ends = 20
!
geometry create shape cylinder  &
cylinder_name = .MODEL_1.ground.Motor2  &
adams_id = 27  &
center_marker = .MODEL_1.ground.MARKER_3  &
angle_extent = 360.0  &
length = 0.2  &
radius = 5.0E-002  &
side_count_for_body = 20  &
segment_count_for_ends = 20
!
geometry create shape cylinder  &
cylinder_name = .MODEL_1.ground.Motor3  &
adams_id = 28  &
center_marker = .MODEL_1.ground.MARKER_4  &
angle_extent = 360.0  &
length = 0.1999956  &
radius = 5.0E-002  &
side_count_for_body = 20  &
segment_count_for_ends = 20
```

```
!  
geometry create shape cylinder &  
    cylinder_name = .MODEL_1.ground.Motor1 &  
    adams_id = 29 &  
    center_marker = .MODEL_1.ground.MARKER_5 &  
    angle_extent = 360.0 &  
    length = 0.1999906 &  
    radius = 5.0E-002 &  
    side_count_for_body = 20 &  
    segment_count_for_ends = 20  
!  
geometry create shape cylinder &  
    cylinder_name = .MODEL_1.ground.Motor6 &  
    adams_id = 30 &  
    center_marker = .MODEL_1.ground.MARKER_6 &  
    angle_extent = 360.0 &  
    length = 0.2000042602 &  
    radius = 5.0E-002 &  
    side_count_for_body = 20 &  
    segment_count_for_ends = 20  
!  
geometry create shape cylinder &  
    cylinder_name = .MODEL_1.ground.Motor5 &  
    adams_id = 31 &  
    center_marker = .MODEL_1.ground.MARKER_7 &
```

```
angle_extent = 360.0  &
length = 0.2  &
radius = 5.0E-002  &
side_count_for_body = 20  &
segment_count_for_ends = 20
!
geometry create shape cylinder  &
    cylinder_name = .MODEL_1.ground.Motor4  &
    adams_id = 32  &
    center_marker = .MODEL_1.ground.MARKER_8  &
    angle_extent = 360.0  &
    length = 0.2000042602  &
    radius = 5.0E-002  &
    side_count_for_body = 20  &
    segment_count_for_ends = 20
!
part attributes  &
    part_name = .MODEL_1.ground  &
    name_visibility = off
!
!----- Plataforma_Movel -----!
!
!
defaults coordinate_system  &
    default_coordinate_system = .MODEL_1.ground
```

```
!  
part create rigid_body name_and_position &  
    part_name = .MODEL_1.Plataforma_Movel &  
    adams_id = 2 &  
    location = 0.0, 0.0, 0.0 &  
    orientation = 0.0d, 0.0d, 0.0d  
!  
defaults coordinate_system &  
    default_coordinate_system = .MODEL_1.Plataforma_Movel  
!  
! ***** Markers for current part *****  
!  
marker create &  
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_2 &  
    adams_id = 2 &  
    location = 0.0, 0.0, -0.45 &  
    orientation = 0.0d, 0.0d, 0.0d  
!  
marker create &  
    marker_name = .MODEL_1.Plataforma_Movel.cm &  
    adams_id = 65 &  
    location = 2.7749703067E-008, 0.0, -0.4471429616 &  
    orientation = 270.0d, 90.0d, 90.0d  
!  
marker create &
```

```
marker_name = .MODEL_1.Plataforma_Movel.MARKER_9  &
adams_id = 9  &
location = 0.29173, 0.40235, -0.45  &
orientation = 270.0d, 0.0d, 0.0d
!

marker create  &
marker_name = .MODEL_1.Plataforma_Movel.MARKER_10  &
adams_id = 10  &
location = 0.29173, -0.40235, -0.45  &
orientation = 270.0d, 0.0d, 0.0d
!

marker create  &
marker_name = .MODEL_1.Plataforma_Movel.MARKER_11  &
adams_id = 11  &
location = 0.20258, -0.45382, -0.45  &
orientation = 149.9999771292d, 180.0d, 0.0d
!

marker create  &
marker_name = .MODEL_1.Plataforma_Movel.MARKER_12  &
adams_id = 12  &
location = -0.49431, -5.147E-002, -0.45  &
orientation = 149.9999771292d, 180.0d, 0.0d
!

marker create  &
marker_name = .MODEL_1.Plataforma_Movel.MARKER_13  &
```

```
    adams_id = 13    &
    location = -0.49431, 5.147E-002, -0.45    &
    orientation = 30.0000228708d, 0.0d, 0.0d
!
marker create    &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_14    &
    adams_id = 14    &
    location = 0.20258, 0.45382, -0.45    &
    orientation = 30.0000228708d, 0.0d, 0.0d
!
marker create    &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_40    &
    adams_id = 40    &
    location = 0.29173, -0.40235, -0.45    &
    orientation = 94.956350251d, 42.9586685434d, 180.0d
!
marker create    &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_42    &
    adams_id = 42    &
    location = 0.29173, 0.40235, -0.45    &
    orientation = 73.2416392566d, 26.1876223141d, 0.0d
!
marker create    &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_44    &
    adams_id = 44    &
```

```
location = 0.20258, 0.45382, -0.45  &
orientation = 226.7586338166d, 26.1881763255d, 180.0d
!
marker create  &
marker_name = .MODEL_1.Plataforma_Movel.MARKER_46  &
adams_id = 46  &
location = -0.49431, 5.147E-002, -0.45  &
orientation = 170.9292629715d, 57.8167932806d, 0.0d
!
marker create  &
marker_name = .MODEL_1.Plataforma_Movel.MARKER_48  &
adams_id = 48  &
location = -0.49431, -5.147E-002, -0.45  &
orientation = 351.8862688373d, 57.6947548425d, 180.0d
!
marker create  &
marker_name = .MODEL_1.Plataforma_Movel.MARKER_50  &
adams_id = 50  &
location = 0.20258, -0.45382, -0.45  &
orientation = 290.9299094419d, 57.8167388276d, 180.0d
!
marker create  &
marker_name = .MODEL_1.Plataforma_Movel.MARKER_64  &
adams_id = 64  &
location = 0.0, 0.0, -0.45  &
```

```
orientation = 180.0d, 90.0d, 180.0d
!
part create rigid_body mass_properties &
    part_name = .MODEL_1.Plataforma_Movel &
    material_type = .MODEL_1.steel
!
! ***** Graphics for current part *****
!
geometry create shape cylinder &
    cylinder_name = .MODEL_1.Plataforma_Movel.Plataforma &
    adams_id = 4 &
    center_marker = .MODEL_1.Plataforma_Movel.MARKER_2 &
    angle_extent = 360.0 &
    length = 1.0E-002 &
    radius = 0.29173 &
    side_count_for_body = 20 &
    segment_count_for_ends = 20
!
geometry create shape link &
    link_name = .MODEL_1.Plataforma_Movel.LINK_33 &
    i_marker = .MODEL_1.Plataforma_Movel.MARKER_9 &
    j_marker = .MODEL_1.Plataforma_Movel.MARKER_10 &
    width = 4.0E-002 &
    depth = 2.0E-002
!
```



```

geometry create shape link &
    link_name = .MODEL_1.Plataforma_Movel.LINK_34 &
    i_marker = .MODEL_1.Plataforma_Movel.MARKER_11 &
    j_marker = .MODEL_1.Plataforma_Movel.MARKER_12 &
    width = 4.0E-002 &
    depth = 2.0E-002
!

geometry create shape link &
    link_name = .MODEL_1.Plataforma_Movel.LINK_35 &
    i_marker = .MODEL_1.Plataforma_Movel.MARKER_13 &
    j_marker = .MODEL_1.Plataforma_Movel.MARKER_14 &
    width = 4.0E-002 &
    depth = 2.0E-002
!

part attributes &
    part_name = .MODEL_1.Plataforma_Movel &
    color = GREEN &
    name_visibility = off
!

!----- Crank2 -----!
!
!

defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!

```

```
part create rigid_body name_and_position &
    part_name = .MODEL_1.Crank2 &
    adams_id = 3 &
    location = 0.0, 0.0, 0.0 &
    orientation = 0.0d, 0.0d, 0.0d
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.Crank2
!
! ***** Markers for current part *****
!
marker create &
    marker_name = .MODEL_1.Crank2.MARKER_15 &
    adams_id = 15 &
    location = -0.14735, 0.65034, 0.0 &
    orientation = 90.0d, 46.6041299865d, 270.0d
!
marker create &
    marker_name = .MODEL_1.Crank2.cm &
    adams_id = 66 &
    location = -7.6869999987E-002, 0.65034, 6.6640 &
    orientation = 270.0d, 133.3958700135d, 90.0d
!
marker create &
    marker_name = .MODEL_1.Crank2.MARKER_29 &
```

```
    adams_id = 29    &
    location = -0.14735, 0.65034, 0.0    &
    orientation = 180.0d, 90.0d, 180.0d
!
marker create    &
    marker_name = .MODEL_1.Crank2.MARKER_55    &
    adams_id = 55    &
    location = -6.39E-003, 0.65034, 0.13328    &
    orientation = 270.0d, 133.3958700135d, 180.0d
!
part create rigid_body mass_properties    &
    part_name = .MODEL_1.Crank2    &
    material_type = .MODEL_1.steel
!
! ***** Graphics for current part *****
!
geometry create shape cylinder    &
    cylinder_name = .MODEL_1.Crank2.CYLINDER_36    &
    adams_id = 36    &
    center_marker = .MODEL_1.Crank2.MARKER_15    &
    angle_extent = 360.0    &
    length = 0.1939929896    &
    radius = 1.5E-002    &
    side_count_for_body = 20    &
    segment_count_for_ends = 20
```

```
!  
part attributes &  
    part_name = .MODEL_1.Crank2 &  
    color = CYAN &  
    name_visibility = off  
!  
!----- limb2 -----!  
!  
!  
defaults coordinate_system &  
    default_coordinate_system = .MODEL_1.ground  
!  
part create rigid_body name_and_position &  
    part_name = .MODEL_1.limb2 &  
    adams_id = 4 &  
    location = 0.0, 0.0, 0.0 &  
    orientation = 0.0d, 0.0d, 0.0d  
!  
defaults coordinate_system &  
    default_coordinate_system = .MODEL_1.limb2  
!  
! ***** Markers for current part *****  
!  
marker create &  
    marker_name = .MODEL_1.limb2.MARKER_16 &
```

```
    adams_id = 16    &
    location = -6.39E-003, 0.65034, 0.13328    &
orientation = 46.7586338166d, 153.8118236745d, 229.87d
!
marker create    &
    marker_name = .MODEL_1.limb2.cm    &
    adams_id = 67    &
    location = 9.8095000002E-002, 0.55208, -0.15836    &
    orientation = 46.7586338166d, 153.8118236745d, 0.0d
!
marker create    &
    marker_name = .MODEL_1.limb2.MARKER_43    &
    adams_id = 43    &
    location = 0.20258, 0.45382, -0.45    &
    orientation = 226.7586338166d, 26.1881763255d, 180.0d
!
marker create    &
    marker_name = .MODEL_1.limb2.MARKER_56    &
    adams_id = 56    &
    location = -6.39E-003, 0.65034, 0.13328    &
    orientation = 270.0d, 133.3958700135d, 180.0d
!
part create rigid_body mass_properties    &
    part_name = .MODEL_1.limb2    &
    material_type = .MODEL_1.steel
```

```
!  
! ***** Graphics for current part *****  
!  
geometry create shape cylinder &  
    cylinder_name = .MODEL_1.limb2.CYLINDER_37 &  
    adams_id = 37 &  
    center_marker = .MODEL_1.limb2.MARKER_16 &  
    angle_extent = 360.0 &  
    length = 0.6500031767 &  
    radius = 1.5E-002 &  
    side_count_for_body = 20 &  
    segment_count_for_ends = 20  
!  
part attributes &  
    part_name = .MODEL_1.limb2 &  
    color = MAGENTA &  
    name_visibility = off  
!  
!----- Crank1 ----!  
!  
!  
defaults coordinate_system &  
    default_coordinate_system = .MODEL_1.ground  
!  
part create rigid_body name_and_position &
```

```

part_name = .MODEL_1.Crank1  &
adams_id = 5  &
location = 0.0, 0.0, 0.0  &
orientation = 0.0d, 0.0d, 0.0d
!
defaults coordinate_system  &
    default_coordinate_system = .MODEL_1.Crank1
!
! ***** Markers for current part *****
!
marker create  &
    marker_name = .MODEL_1.Crank1.MARKER_17  &
    adams_id = 17  &
    location = 0.63689, 0.19756, 0.0  &
orientation = 29.9954980575d, 122.3437522524d, 227.15d
!
marker create  &
    marker_name = .MODEL_1.Crank1.cm  &
    adams_id = 68  &
    location = 0.6778597794, 0.1265853822, -5.183E-002  &
    orientation = 209.9954980575d, 57.6562477476d, 90.0d
!
marker create  &
    marker_name = .MODEL_1.Crank1.MARKER_31  &
    adams_id = 31  &

```

```
location = 0.63689, 0.19756, 0.0  &
orientation = 120.0007277808d, 90.0d, 270.0d
!
marker create  &
    marker_name = .MODEL_1.Crank1.MARKER_53  &
    adams_id = 53  &
    location = 0.71883, 5.561E-002, -0.10379  &
    orientation = 30.0024917749d, 133.393966442d, 0.0d
!
part create rigid_body mass_properties  &
    part_name = .MODEL_1.Crank1  &
    material_type = .MODEL_1.steel
!
! ***** Graphics for current part *****
!
geometry create shape cylinder  &
    cylinder_name = .MODEL_1.Crank1.CYLINDER_38  &
    adams_id = 38  &
    center_marker = .MODEL_1.Crank1.MARKER_17  &
    angle_extent = 360.0  &
    length = 0.1939998064  &
    radius = 1.5E-002  &
    side_count_for_body = 20  &
    segment_count_for_ends = 20
!
```



```

part attributes &
    part_name = .MODEL_1.Crank1 &
    color = RED &
    name_visibility = off
!
!----- limb1 -----!
!
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
part create rigid_body name_and_position &
    part_name = .MODEL_1.limb1 &
    adams_id = 6 &
    location = 0.0, 0.0, 0.0 &
    orientation = 0.0d, 0.0d, 0.0d
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.limb1
!
! ***** Markers for current part *****
!
marker create &
marker_name = .MODEL_1.limb1.MARKER_18 &
adams_id = 18 &

```

```
location = 0.71883, 5.561E-002, -0.10379  &
orientation = 230.9286730075d, 122.1831904104d, 66.6d
!
marker create  &
marker_name = .MODEL_1.limb1.cm  &
adams_id = 69  &
location = 0.5052810842, 0.2289791198, -0.2768941211&
orientation = 50.9286730075d, 57.8168095896d, 0.0d
!
marker create  &
    marker_name = .MODEL_1.limb1.MARKER_41  &
    adams_id = 41  &
    location = 0.29173, 0.40235, -0.45  &
    orientation = 73.2416392566d, 26.1876223141d, 0.0d
!
marker create  &
    marker_name = .MODEL_1.limb1.MARKER_54  &
    adams_id = 54  &
    location = 0.71883, 5.561E-002, -0.10379  &
    orientation = 30.0024917749d, 133.393966442d, 0.0d
!
part create rigid_body mass_properties  &
    part_name = .MODEL_1.limb1  &
    material_type = .MODEL_1.steel
!
```

```

! ***** Graphics for current part *****
!
geometry create shape cylinder &
    cylinder_name = .MODEL_1.limb1.CYLINDER_39&
    adams_id = 39 &
    center_marker = .MODEL_1.limb1.MARKER_18 &
    angle_extent = 360.0 &
    length = 0.6500000857 &
    radius = 1.5E-002 &
    side_count_for_body = 20 &
    segment_count_for_ends = 20
!
part attributes &
    part_name = .MODEL_1.limb1 &
    color = GREEN &
    name_visibility = off
!
!----- Crank6 -----!
!
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
part create rigid_body name_and_position &
    part_name = .MODEL_1.Crank6 &

```

```
    adams_id = 7    &
    location = 0.0, 0.0, 0.0    &
    orientation = 0.0d, 0.0d, 0.0d
!
defaults coordinate_system    &
    default_coordinate_system = .MODEL_1.Crank6
!
! ***** Markers for current part *****
!
marker create    &
    marker_name = .MODEL_1.Crank6.MARKER_19    &
    adams_id = 19    &
    location = 0.63689, -0.19756, 0.0    &
orientation = 329.9975082251d, 46.606033558d, 40.0458d
!
marker create    &
    marker_name = .MODEL_1.Crank6.cm    &
    adams_id = 70    &
location = 0.6016466973,-0.2585970604,6.66E-002&
    orientation = 149.9975082251d, 133.393966442d, 90.0d
!
marker create    &
    marker_name = .MODEL_1.Crank6.MARKER_79    &
    adams_id = 79    &
    location = 0.63689, -0.19756, 0.0    &
```

```
orientation = 59.9992722192d, 90.0d, 270.0d
!
marker create &
    marker_name = .MODEL_1.Crank6.MARKER_51 &
    adams_id = 51 &
    location = 0.5664, -0.31964, 0.13328 &
    orientation = 209.9999053376d, 97.6129683539d, 180.0d
!
part create rigid_body mass_properties &
    part_name = .MODEL_1.Crank6 &
    material_type = .MODEL_1.steel
!
! ***** Graphics for current part *****
!
geometry create shape cylinder &
    cylinder_name = .MODEL_1.Crank6.CYLINDER_40 &
    adams_id = 40 &
    center_marker = .MODEL_1.Crank6.MARKER_19 &
    angle_extent = 360.0 &
    length = 0.1939904637 &
    radius = 1.5E-002 &
    side_count_for_body = 20 &
    segment_count_for_ends = 20
!
part attributes &
```

```
part_name = .MODEL_1.Crank6  &
color = MAIZE  &
name_visibility = off
!
!----- limb6 -----!
!
!
defaults coordinate_system  &
    default_coordinate_system = .MODEL_1.ground
!
part create rigid_body name_and_position  &
    part_name = .MODEL_1.limb6  &
    adams_id = 8  &
    location = 0.0, 0.0, 0.0  &
    orientation = 0.0d, 0.0d, 0.0d
!
defaults coordinate_system  &
    default_coordinate_system = .MODEL_1.limb6
!
! ***** Markers for current part *****
!
marker create  &
    marker_name = .MODEL_1.limb6.MARKER_20  &
    adams_id = 20  &
    location = 0.5664, -0.31964, 0.13328  &
```

```
orientation = 286.7583607434d, 153.8123776859d, 105.12d
!
marker create &
    marker_name = .MODEL_1.limb6.cm &
    adams_id = 71 &
    location = 0.4290648925, -0.3609950324, -0.1583 &
    orientation = 106.7583607434d, 26.1876223141d, 270.0d
!
marker create &
    marker_name = .MODEL_1.limb6.MARKER_39 &
    adams_id = 39 &
    location = 0.29173, -0.40235, -0.45 &
    orientation = 94.956350251d, 42.9586685434d, 180.0d
!
marker create &
    marker_name = .MODEL_1.limb6.MARKER_52 &
    adams_id = 52 &
    location = 0.5664, -0.31964, 0.13328 &
    orientation = 209.9999053376d, 97.6129683539d, 180.0d
!
part create rigid_body mass_properties &
    part_name = .MODEL_1.limb6 &
    material_type = .MODEL_1.steel
!
! ***** Graphics for current part *****
```

```
!  
geometry create shape cylinder &  
    cylinder_name = .MODEL_1.limb6.CYLINDER_41 &  
    adams_id = 41 &  
    center_marker = .MODEL_1.limb6.MARKER_20 &  
    angle_extent = 360.0 &  
    length = 0.6500005945 &  
    radius = 1.5E-002 &  
    side_count_for_body = 20 &  
    segment_count_for_ends = 20  
!  
part attributes &  
    part_name = .MODEL_1.limb6 &  
    color = CYAN &  
    name_visibility = off  
!  
!----- Crank5 -----!  
!  
!  
defaults coordinate_system &  
    default_coordinate_system = .MODEL_1.ground  
!  
part create rigid_body name_and_position &  
    part_name = .MODEL_1.Crank5 &  
    adams_id = 9 &
```



```
location = 0.0, 0.0, 0.0  &
orientation = 0.0d, 0.0d, 0.0d
!
defaults coordinate_system  &
    default_coordinate_system = .MODEL_1.Crank5
!
! ***** Markers for current part *****
!
marker create  &
    marker_name = .MODEL_1.Crank5.MARKER_21  &
    adams_id = 21  &
    location = -0.14735, -0.65034, 0.0  &
    orientation = 270.0d, 122.3441167062d, 90.0d
!
marker create  &
    marker_name = .MODEL_1.Crank5.cm  &
    adams_id = 72  &
    location = -0.2293, -0.65034, -5.18949999996E-002&
    orientation = 270.0d, 122.3441167062d, 90.0d
!
marker create  &
    marker_name = .MODEL_1.Crank5.MARKER_35  &
    adams_id = 35  &
    location = -0.14735, -0.65034, 0.0  &
    orientation = 0.0d, 90.0d, 0.0d
```

```
!  
marker create &  
    marker_name = .MODEL_1.Crank5.MARKER_61 &  
    adams_id = 61 &  
    location = -0.31125, -0.65034, -0.10379 &  
    orientation = 90.0d, 57.6558832938d, 0.0d  
!  
part create rigid_body mass_properties &  
    part_name = .MODEL_1.Crank5 &  
    material_type = .MODEL_1.steel  
!  
! ***** Graphics for current part *****  
!  
geometry create shape cylinder &  
    cylinder_name = .MODEL_1.Crank5.CYLINDER_42 &  
    adams_id = 42 &  
    center_marker = .MODEL_1.Crank5.MARKER_21 &  
    angle_extent = 360.0 &  
    length = 0.1939989023 &  
    radius = 1.5E-002 &  
    side_count_for_body = 20 &  
    segment_count_for_ends = 20  
!  
part attributes &  
    part_name = .MODEL_1.Crank5 &
```

```

    color = MAGENTA    &
    name_visibility = off
!
!----- limb5 -----!
!
!
defaults coordinate_system    &
    default_coordinate_system = .MODEL_1.ground
!
part create rigid_body name_and_position    &
    part_name = .MODEL_1.limb5    &
    adams_id = 10    &
    location = 0.0, 0.0, 0.0    &
    orientation = 0.0d, 0.0d, 0.0d
!
defaults coordinate_system    &
    default_coordinate_system = .MODEL_1.limb5
!
! ***** Markers for current part *****
!
marker create    &
    marker_name = .MODEL_1.limb5.MARKER_22    &
    adams_id = 22    &
    location = -0.31125, -0.65034, -0.10379    &
    orientation = 110.9299094419d, 122.1832611724d, 281.3d

```

```
!  
marker create &  
    marker_name = .MODEL_1.limb5.cm &  
    adams_id = 73 &  
    location = -5.4335000002E-002, -0.55208, -0.276&  
    orientation = 290.9299094419d, 57.8167388276d, 90.0d  
!  
marker create &  
    marker_name = .MODEL_1.limb5.MARKER_49 &  
    adams_id = 49 &  
    location = 0.20258, -0.45382, -0.45 &  
    orientation = 290.9299094419d, 57.8167388276d, 180.0d  
!  
marker create &  
    marker_name = .MODEL_1.limb5.MARKER_62 &  
    adams_id = 62 &  
    location = -0.31125, -0.65034, -0.10379 &  
    orientation = 90.0d, 57.6558832938d, 0.0d  
!  
part create rigid_body mass_properties &  
    part_name = .MODEL_1.limb5 &  
    material_type = .MODEL_1.steel  
!  
! ***** Graphics for current part *****  
!
```

```

geometry create shape cylinder &
    cylinder_name = .MODEL_1.limb5.CYLINDER_43 &
    adams_id = 43 &
    center_marker = .MODEL_1.limb5.MARKER_22 &
    angle_extent = 360.0 &
    length = 0.6500021103 &
    radius = 1.5E-002 &
    side_count_for_body = 20 &
    segment_count_for_ends = 20
!
part attributes &
    part_name = .MODEL_1.limb5 &
    color = RED &
    name_visibility = off
!
!----- Crank4 ----!
!
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
part create rigid_body name_and_position &
    part_name = .MODEL_1.Crank4 &
    adams_id = 11 &
    location = 0.0, 0.0, 0.0 &

```

```
orientation = 0.0d, 0.0d, 0.0d
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.Crank4
!
! ***** Markers for current part *****
!
marker create &
    marker_name = .MODEL_1.Crank4.MARKER_23 &
    adams_id = 23 &
    location = -0.48953, -0.45278, 0.0 &
    orientation = 209.9989718535d, 46.6050190096d, 139.95d
!
marker create &
    marker_name = .MODEL_1.Crank4.cm &
    adams_id = 74 &
    location = -0.5247709307, -0.3917383878, 6.6646E-002 &
    orientation = 209.9989718535d, 46.6050190096d, 90.0d
!
marker create &
    marker_name = .MODEL_1.Crank4.MARKER_37 &
    adams_id = 37 &
    location = -0.48953, -0.45278, 0.0 &
    orientation = 299.9992953853d, 90.0d, 90.0d
!
```

```
marker create &
    marker_name = .MODEL_1.Crank4.MARKER_60 &
    adams_id = 60 &
    location = -0.56001, -0.3307, 0.13328 &
    orientation = 149.9995784755d, 58.0644131243d, 0.0d
!
part create rigid_body mass_properties &
    part_name = .MODEL_1.Crank4 &
    material_type = .MODEL_1.steel
!
! ***** Graphics for current part *****
!
geometry create shape cylinder &
    cylinder_name = .MODEL_1.Crank4.CYLINDER_44 &
    adams_id = 44 &
    center_marker = .MODEL_1.Crank4.MARKER_23 &
    angle_extent = 360.0 &
    length = 0.1940012969 &
    radius = 1.5E-002 &
    side_count_for_body = 20 &
    segment_count_for_ends = 20
!
part attributes &
    part_name = .MODEL_1.Crank4 &
    color = GREEN &
```

```
name_visibility = off
!
!----- limb4 -----!
!
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
part create rigid_body name_and_position &
    part_name = .MODEL_1.limb4 &
    adams_id = 12 &
    location = 0.0, 0.0, 0.0 &
    orientation = 0.0d, 0.0d, 0.0d
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.limb4
!
! ***** Markers for current part *****
!
marker create &
    marker_name = .MODEL_1.limb4.MARKER_24 &
    adams_id = 24 &
    location = -0.56001, -0.3307, 0.13328 &
    orientation = 166.7597088237d, 153.8121943277d, 345
!
```



```

marker create &
    marker_name = .MODEL_1.limb4.cm &
    adams_id = 75 &
location = -0.527159992, -0.1910849658, -0.158314&
orientation = 166.7597088237d, 153.8121943277d, 90.0d
!
marker create &
    marker_name = .MODEL_1.limb4.MARKER_47 &
    adams_id = 47 &
    location = -0.49431, -5.147E-002, -0.45 &
orientation = 351.8862688373d, 57.6947548425d, 180.0d
!
marker create &
    marker_name = .MODEL_1.limb4.MARKER_59 &
    adams_id = 59 &
    location = -0.56001, -0.3307, 0.13328 &
orientation = 149.9995784755d, 58.0644131243d, 0.0d
!
part create rigid_body mass_properties &
    part_name = .MODEL_1.limb4 &
    material_type = .MODEL_1.steel
!
! ***** Graphics for current part *****
!
geometry create shape cylinder &

```

```
cylinder_name = .MODEL_1.limb4.CYLINDER_45  &
adams_id = 45  &
center_marker = .MODEL_1.limb4.MARKER_24  &
angle_extent = 360.0  &
length = 0.6500012678  &
radius = 1.5E-002  &
side_count_for_body = 20  &
segment_count_for_ends = 20
!
part attributes  &
    part_name = .MODEL_1.limb4  &
    color = MAIZE  &
    name_visibility = off
!
!----- Crank3 -----!
!
!
defaults coordinate_system  &
    default_coordinate_system = .MODEL_1.ground
!
part create rigid_body name_and_position  &
    part_name = .MODEL_1.Crank3  &
    adams_id = 13  &
    location = 0.0, 0.0, 0.0  &
    orientation = 0.0d, 0.0d, 0.0d
```

```
!  
defaults coordinate_system &  
    default_coordinate_system = .MODEL_1.Crank3  
!  
! ***** Markers for current part *****  
!  
marker create &  
    marker_name = .MODEL_1.Crank3.MARKER_25 &  
    adams_id = 25 &  
    location = -0.48953, 0.45278, 0.0 &  
orientation = 149.9997266839d, 122.3443306798d, 312.8  
!  
marker create &  
    marker_name = .MODEL_1.Crank3.cm &  
    adams_id = 76 &  
location = -0.448555, 0.52375, -5.18949999991E-002 &  
orientation = 149.9997266839d, 122.3443306798d, 90.0d  
!  
marker create &  
    marker_name = .MODEL_1.Crank3.MARKER_27 &  
    adams_id = 27 &  
    location = -0.48953, 0.45278, 0.0 &  
    orientation = 240.0007046147d, 90.0d, 90.0d  
!  
marker create &
```

```
marker_name = .MODEL_1.Crank3.MARKER_57  &
adams_id = 57  &
location = -0.40758, 0.59472, -0.10379  &
orientation = 329.9997266839d, 57.6556693202d, 180.0d
!
part create rigid_body mass_properties  &
    part_name = .MODEL_1.Crank3  &
    material_type = .MODEL_1.steel
!
! ***** Graphics for current part *****
!
geometry create shape cylinder  &
    cylinder_name = .MODEL_1.Crank3.CYLINDER_46  &
    adams_id = 46  &
    center_marker = .MODEL_1.Crank3.MARKER_25  &
    angle_extent = 360.0  &
    length = 0.1939977582  &
    radius = 1.5E-002  &
    side_count_for_body = 20  &
    segment_count_for_ends = 20
!
part attributes  &
    part_name = .MODEL_1.Crank3  &
    color = CYAN  &
    name_visibility = off
```

```

!
!----- limb3 -----!
!
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
part create rigid_body name_and_position &
    part_name = .MODEL_1.limb3 &
    adams_id = 14 &
    location = 0.0, 0.0, 0.0 &
    orientation = 0.0d, 0.0d, 0.0d
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.limb3
!
! ***** Markers for current part *****
!
marker create &
    marker_name = .MODEL_1.limb3.MARKER_26 &
    adams_id = 26 &
    location = -0.40758, 0.59472, -0.10379 &
    orientation = 350.9292629715d, 122.1832067194d, 163.31
!
marker create &

```

```
marker_name = .MODEL_1.limb3.cm  &
adams_id = 77  &
location = -0.450945, 0.323095, -0.276895  &
orientation = 170.9292629715d, 57.8167932806d, 90.0d
!
marker create  &
marker_name = .MODEL_1.limb3.MARKER_45  &
adams_id = 45  &
location = -0.49431, 5.147E-002, -0.45  &
orientation = 170.9292629715d, 57.8167932806d, 0.0d
!
marker create  &
marker_name = .MODEL_1.limb3.MARKER_58  &
adams_id = 58  &
location = -0.40758, 0.59472, -0.10379  &
orientation = 329.9997266839d, 57.6556693202d, 180.0d
!
part create rigid_body mass_properties  &
part_name = .MODEL_1.limb3  &
material_type = .MODEL_1.steel
!
! ***** Graphics for current part *****
!
geometry create shape cylinder  &
cylinder_name = .MODEL_1.limb3.CYLINDER_47  &
```

```

    adams_id = 47    &
    center_marker = .MODEL_1.limb3.MARKER_26    &
    angle_extent = 360.0    &
    length = 0.6500030919    &
    radius = 1.5E-002    &
    side_count_for_body = 20    &
    segment_count_for_ends = 20
!
part attributes    &
    part_name = .MODEL_1.limb3    &
    color = MAGENTA    &
    name_visibility = off
!
!----- Joints -----!
!
!
constraint create joint revolute    &
    joint_name = .MODEL_1.JuntaR3    &
    adams_id = 1    &
    i_marker_name = .MODEL_1.Crank3.MARKER_27    &
    j_marker_name = .MODEL_1.ground.MARKER_28
!
constraint attributes    &
    constraint_name = .MODEL_1.JuntaR3    &
    name_visibility = off

```

```
!  
constraint create joint revolute &  
    joint_name = .MODEL_1.JuntaR2 &  
    adams_id = 2 &  
    i_marker_name = .MODEL_1.Crank2.MARKER_29 &  
    j_marker_name = .MODEL_1.ground.MARKER_30  
!  
constraint attributes &  
    constraint_name = .MODEL_1.JuntaR2 &  
    name_visibility = off  
!  
constraint create joint revolute &  
    joint_name = .MODEL_1.JuntaR1 &  
    adams_id = 3 &  
    i_marker_name = .MODEL_1.Crank1.MARKER_31 &  
    j_marker_name = .MODEL_1.ground.MARKER_32  
!  
constraint attributes &  
    constraint_name = .MODEL_1.JuntaR1 &  
    name_visibility = off  
!  
constraint create joint revolute &  
    joint_name = .MODEL_1.Junta_R6 &  
    adams_id = 20 &  
    i_marker_name = .MODEL_1.ground.MARKER_78 &
```



```
j_marker_name = .MODEL_1.Crank6.MARKER_79
!
constraint attributes &
    constraint_name = .MODEL_1.Junta_R6 &
    name_visibility = off
!
constraint create joint revolute &
    joint_name = .MODEL_1.JuntaR5 &
    adams_id = 5 &
    i_marker_name = .MODEL_1.Crank5.MARKER_35 &
    j_marker_name = .MODEL_1.ground.MARKER_36
!
constraint attributes &
    constraint_name = .MODEL_1.JuntaR5 &
    name_visibility = off
!
constraint create joint revolute &
    joint_name = .MODEL_1.JuntaR4 &
    adams_id = 6 &
    i_marker_name = .MODEL_1.Crank4.MARKER_37 &
    j_marker_name = .MODEL_1.ground.MARKER_38
!
constraint attributes &
    constraint_name = .MODEL_1.JuntaR4 &
    name_visibility = off
```

```
!  
constraint create joint spherical &  
    joint_name = .MODEL_1.JuntaS6 &  
    adams_id = 7 &  
    i_marker_name = .MODEL_1.limb6.MARKER_39 &  
    j_marker_name = .MODEL_1.Plataforma_Movel.MARKER_40  
!  
constraint attributes &  
    constraint_name = .MODEL_1.JuntaS6 &  
    name_visibility = off  
!  
constraint create joint spherical &  
    joint_name = .MODEL_1.JuntaS1 &  
    adams_id = 8 &  
    i_marker_name = .MODEL_1.limb1.MARKER_41 &  
    j_marker_name = .MODEL_1.Plataforma_Movel.MARKER_42  
!  
constraint attributes &  
    constraint_name = .MODEL_1.JuntaS1 &  
    name_visibility = off  
!  
constraint create joint spherical &  
    joint_name = .MODEL_1.JuntaS2 &  
    adams_id = 9 &  
    i_marker_name = .MODEL_1.limb2.MARKER_43 &
```

```
j_marker_name = .MODEL_1.Plataforma_Movel.MARKER_44
!
constraint attributes &
    constraint_name = .MODEL_1.JuntaS2 &
    name_visibility = off
!
constraint create joint spherical &
    joint_name = .MODEL_1.JuntaS3 &
    adams_id = 10 &
    i_marker_name = .MODEL_1.limb3.MARKER_45 &
    j_marker_name = .MODEL_1.Plataforma_Movel.MARKER_46
!
constraint attributes &
    constraint_name = .MODEL_1.JuntaS3 &
    name_visibility = off
!
constraint create joint spherical &
    joint_name = .MODEL_1.JuntaS4 &
    adams_id = 11 &
    i_marker_name = .MODEL_1.limb4.MARKER_47 &
    j_marker_name = .MODEL_1.Plataforma_Movel.MARKER_48
!
constraint attributes &
    constraint_name = .MODEL_1.JuntaS4 &
    name_visibility = off
```

```
!  
constraint create joint spherical &  
    joint_name = .MODEL_1.JuntaS5 &  
    adams_id = 12 &  
    i_marker_name = .MODEL_1.limb5.MARKER_49 &  
    j_marker_name = .MODEL_1.Plataforma_Movel.MARKER_50  
!  
constraint attributes &  
    constraint_name = .MODEL_1.JuntaS5 &  
    name_visibility = off  
!  
constraint create joint spherical &  
    joint_name = .MODEL_1.JOINT_13 &  
    adams_id = 13 &  
    i_marker_name = .MODEL_1.Crank6.MARKER_51 &  
    j_marker_name = .MODEL_1.limb6.MARKER_52  
!  
constraint attributes &  
    constraint_name = .MODEL_1.JOINT_13 &  
    name_visibility = off  
!  
constraint create joint spherical &  
    joint_name = .MODEL_1.JOINT_14 &  
    adams_id = 14 &  
    i_marker_name = .MODEL_1.Crank1.MARKER_53 &
```

```
j_marker_name = .MODEL_1.limb1.MARKER_54
!
constraint attributes &
    constraint_name = .MODEL_1.JOINT_14 &
    name_visibility = off
!
constraint create joint spherical &
    joint_name = .MODEL_1.JOINT_15 &
    adams_id = 15 &
    i_marker_name = .MODEL_1.Crank2.MARKER_55 &
    j_marker_name = .MODEL_1.limb2.MARKER_56
!
constraint attributes &
    constraint_name = .MODEL_1.JOINT_15 &
    name_visibility = off
!
constraint create joint spherical &
    joint_name = .MODEL_1.JOINT_16 &
    adams_id = 16 &
    i_marker_name = .MODEL_1.Crank3.MARKER_57 &
    j_marker_name = .MODEL_1.limb3.MARKER_58
!
constraint attributes &
    constraint_name = .MODEL_1.JOINT_16 &
    name_visibility = off
```

```
!  
constraint create joint spherical &  
    joint_name = .MODEL_1.JOINT_17 &  
    adams_id = 17 &  
    i_marker_name = .MODEL_1.limb4.MARKER_59 &  
    j_marker_name = .MODEL_1.Crank4.MARKER_60  
!  
constraint attributes &  
    constraint_name = .MODEL_1.JOINT_17 &  
    name_visibility = off  
!  
constraint create joint spherical &  
    joint_name = .MODEL_1.JOINT_18 &  
    adams_id = 18 &  
    i_marker_name = .MODEL_1.Crank5.MARKER_61 &  
    j_marker_name = .MODEL_1.limb5.MARKER_62  
!  
constraint attributes &  
    constraint_name = .MODEL_1.JOINT_18 &  
    name_visibility = off  
!  
constraint create joint revolute &  
    joint_name = .MODEL_1.Atuador_rotacao &  
    adams_id = 19 &  
    i_marker_name = .MODEL_1.ground.MARKER_63 &
```

```

j_marker_name = .MODEL_1.Plataforma_Movel.MARKER_64
!
constraint attributes &
    constraint_name = .MODEL_1.Atuador_rotacao &
    name_visibility = off
!
!----- Forces -----!
!
!
!----- Simulation Scripts ----!
!
!
simulation script create &
    sim_script_name = .MODEL_1.Last_Sim &
    commands = &
"simulation single_run transient type=auto_select
end_time=5.0 number_of_steps=5000 model_name=.MODEL_1
    initial_static=no"
!
!----- Adams/View UDE Instances -----!
!
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!

```

```
undo begin_block suppress = yes
!
ude create instance &
    instance_name = .MODEL_1.general_motion_1 &
definition_name = .MDI.Constraints.general_motion &
    location = 0.0, 0.0, 0.0 &
    orientation = 0.0, 0.0, 0.0
!
!--- Adams/View UDE Instance -----!
!
!
variable modify &
variable_name = .MODEL_1.general_motion_1.i_marker&
    object_value = (.MODEL_1.ground.MARKER_63)
!
variable modify &
variable_name = .MODEL_1.general_motion_1.j_marker &
    object_value = (.MODEL_1.Plataforma_Movel.MARKER_64)
!
variable modify &
variable_name = .MODEL_1.general_motion_1.constraint&
    object_value = (.MODEL_1.Atuador_rotacao)
!
variable modify &
    variable_name = .MODEL_1.general_motion_1.t1_type&
```



```
integer_value = 0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.t2_type&
integer_value = 0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.t3_type&
integer_value = 0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.r1_type&
integer_value = 0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.r2_type&
integer_value = 0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.r3_type&
integer_value = 1
!
variable modify &
variable_name = .MODEL_1.general_motion_1.t1_func&
string_value = "0 * time"
```

```
!  
variable modify &  
variable_name = .MODEL_1.general_motion_1.t2_func&  
    string_value = "0 * time"  
!  
variable modify &  
variable_name = .MODEL_1.general_motion_1.t3_func&  
    string_value = "0 * time"  
!  
variable modify &  
variable_name = .MODEL_1.general_motion_1.r1_func&  
    string_value = "0 * time"  
!  
variable modify &  
variable_name = .MODEL_1.general_motion_1.r2_func&  
    string_value = "0 * time"  
!  
variable modify &  
variable_name = .MODEL_1.general_motion_1.r3_func&  
    string_value = "1 * time"  
!  
variable modify &  
variable_name = .MODEL_1.general_motion_1.t1_ic_disp&  
    real_value = 0.0  
!
```

```
variable modify &
  variable_name = .MODEL_1.general_motion_1.t2_ic_disp&
  real_value = 0.0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.t3_ic_disp&
  real_value = 0.0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.r1_ic_disp&
  real_value = 0.0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.r2_ic_disp&
  real_value = 0.0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.r3_ic_disp&
  real_value = 0.0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.t1_ic_velo&
  real_value = 0.0
!
variable modify &
```

```
variable_name = .MODEL_1.general_motion_1.t2_ic_velo&
    real_value = 0.0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.t3_ic_velo&
    real_value = 0.0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.r1_ic_velo&
    real_value = 0.0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.r2_ic_velo&
    real_value = 0.0
!
variable modify &
variable_name = .MODEL_1.general_motion_1.r3_ic_velo&
    real_value = 0.0
!
ude modify instance &
    instance_name = .MODEL_1.general_motion_1
!
undo end_block
!
!----- Accgrav -----!
```

```

!
!
force create body gravitational &
    gravity_field_name = gravity &
    x_component_gravity = 0.0 &
    y_component_gravity = 0.0 &
    z_component_gravity = 9.80665
!
!----- Analysis settings -----!
!
!
!----- Function definitions -----!
!
!
!----- Adams/View UDE Instance -----!
!
!
ude modify instance &
    instance_name = .MODEL_1.general_motion_1
!
!----- Expression definitions -----!
!
!
defaults coordinate_system &
    default_coordinate_system = ground

```

```
!  
marker modify &  
    marker_name = .MODEL_1.ground.MARKER_63 &  
    location = &  
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.TCP))  
!  
marker modify &  
    marker_name = .MODEL_1.ground.MARKER_1 &  
    location = &  
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.Origem))  
!  
marker modify &  
    marker_name = .MODEL_1.ground.MARKER_3 &  
    location = &  
        (LOC_RELATIVE_TO({0, 0, 0},  
            .MODEL_1.ground.A2)) &  
    orientation = &  
(ORI_ALONG_AXIS(.MODEL_1.ground.A2,  
    .MODEL_1.ground.A2_2,"Z"))  
!  
marker modify &  
    marker_name = .MODEL_1.ground.MARKER_4 &  
    location = &  
        (LOC_RELATIVE_TO({0, 0, 0},  
            .MODEL_1.ground.A3)) &
```

```
orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A3,
.MODEL_1.ground.A3_2,"Z"))
!
marker modify &
marker_name = .MODEL_1.ground.MARKER_5 &
location = &
(LOC_RELATIVE_TO({0, 0, 0},
.MODEL_1.ground.A1)) &
orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A1,
.MODEL_1.ground.A1_2,"Z"))
!
marker modify &
marker_name = .MODEL_1.ground.MARKER_6 &
location = &
(LOC_RELATIVE_TO({0, 0, 0},
.MODEL_1.ground.A6)) &
orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A6,
.MODEL_1.ground.A6_2,"Z"))
!
marker modify &
marker_name = .MODEL_1.ground.MARKER_7 &
location = &
```

```

        (LOC_RELATIVE_TO({0, 0, 0},
            .MODEL_1.ground.A5)) &
orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A5,
    .MODEL_1.ground.A5_2, "Z"))
!
marker modify &
    marker_name = .MODEL_1.ground.MARKER_8 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
            .MODEL_1.ground.A4)) &
orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A4,
    .MODEL_1.ground.A4_2, "Z"))
!
marker modify &
    marker_name = .MODEL_1.ground.MARKER_28 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
            .MODEL_1.ground.A3)) &
orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A3,
    .MODEL_1.ground.A3_2, "Z"))
!
marker modify &

```



```

marker_name = .MODEL_1.ground.MARKER_30  &
location =  &
    (LOC_RELATIVE_TO({0, 0, 0},
    .MODEL_1.ground.A2))  &
orientation =  &
(ORI_ALONG_AXIS(.MODEL_1.ground.A2,
.MODEL_1.ground.A2_2,"Z"))
!
marker modify  &
marker_name = .MODEL_1.ground.MARKER_32  &
location =  &
    (LOC_RELATIVE_TO({0, 0, 0},
    .MODEL_1.ground.A1))  &
orientation =  &
(ORI_ALONG_AXIS(.MODEL_1.ground.A1,
.MODEL_1.ground.A1_2,"Z"))
!
marker modify  &
marker_name = .MODEL_1.ground.MARKER_36  &
location =  &
    (LOC_RELATIVE_TO({0, 0, 0},
    .MODEL_1.ground.A5))  &
orientation =  &
(ORI_ALONG_AXIS(.MODEL_1.ground.A5,
.MODEL_1.ground.A5_2,"Z"))

```

```
!  
marker modify &  
    marker_name = .MODEL_1.ground.MARKER_38 &  
    location = &  
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.A4)) &  
    orientation = &  
(ORI_ALONG_AXIS(.MODEL_1.ground.A4,  
.MODEL_1.ground.A4_2,"Z"))  
!  
geometry modify shape cylinder &  
    cylinder_name = .MODEL_1.ground.Base &  
    length = (1.0cm) &  
    radius = (0.636885m)  
!  
geometry modify shape cylinder &  
    cylinder_name = .MODEL_1.ground.Motor2 &  
    length = (0.2meter) &  
    radius = (5.0cm)  
!  
geometry modify shape cylinder &  
    cylinder_name = .MODEL_1.ground.Motor3 &  
    length = (0.1999956meter) &  
    radius = (5.0cm)  
!  
geometry modify shape cylinder &
```

```

    cylinder_name = .MODEL_1.ground.Motor1  &
    length = (0.1999906meter)  &
    radius = (5.0cm)
!
geometry modify shape cylinder  &
    cylinder_name = .MODEL_1.ground.Motor6  &
    length = (0.2000042602meter)  &
    radius = (5.0cm)
!
geometry modify shape cylinder  &
    cylinder_name = .MODEL_1.ground.Motor5  &
    length = (0.2meter)  &
    radius = (5.0cm)
!
geometry modify shape cylinder  &
    cylinder_name = .MODEL_1.ground.Motor4  &
    length = (0.2000042602meter)  &
    radius = (5.0cm)
!
marker modify  &
    marker_name = .MODEL_1.ground.MARKER_78  &
    location =  &
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.A6))  &
    orientation =  &
(ORI_ALONG_AXIS(.MODEL_1.ground.A6,

```

```

.MODEL_1.ground.A6_2,"Z"))
!
material modify &
    material_name = .MODEL_1.steel &
    youngs_modulus = (2.07E+011 (Newton/meter**2)) &
    density = (7801.0 (kg/meter**3))
!
marker modify &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_2&
    location = &
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.TCP))&
    relative_to = .MODEL_1.Plataforma_Movel
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_9&
    location = &
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C1)) &
    orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.C1,
.MODEL_1.ground.C6, "X"))&
    relative_to = .MODEL_1.Plataforma_Movel
!

```

```

defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!

marker modify &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_10&
    location = &
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C6))&
    orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.C1,
    .MODEL_1.ground.C6, "X"))&
    relative_to = .MODEL_1.Plataforma_Movel
!

defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!

marker modify &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_11&
    location = &
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C5))&
    orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.C5,
    .MODEL_1.ground.C4, "X")) &
    relative_to = .MODEL_1.Plataforma_Movel
!

defaults coordinate_system &

```

```

    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_12&
    location = &
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C4)) &
    orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.C5,
.MODEL_1.ground.C4,"X"))&
    relative_to = .MODEL_1.Plataforma_Movel
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
marker_name = .MODEL_1.Plataforma_Movel.MARKER_13&
    location = &
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C3)) &
    orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.C3,
.MODEL_1.ground.C2,"X")) &
    relative_to = .MODEL_1.Plataforma_Movel
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground

```

```

!
marker modify  &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_14  &
    location =  &
    (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C2))  &
    orientation =  &
    (ORI_ALONG_AXIS(.MODEL_1.ground.C3,
.MODEL_1.ground.C2,"X"))&
    relative_to = .MODEL_1.Plataforma_Movel
!
defaults coordinate_system  &
    default_coordinate_system = .MODEL_1.ground
!
marker modify  &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_40&
    location =  &
    (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C6))&
    relative_to = .MODEL_1.Plataforma_Movel
!
defaults coordinate_system  &
    default_coordinate_system = .MODEL_1.ground
!
marker modify  &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_42&
    location =  &

```

```
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C1))&
relative_to = .MODEL_1.Plataforma_Movel
!
defaults coordinate_system &
default_coordinate_system = .MODEL_1.ground
!
marker modify &
marker_name = .MODEL_1.Plataforma_Movel.MARKER_44&
location = &
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C2))&
relative_to = .MODEL_1.Plataforma_Movel
!
defaults coordinate_system &
default_coordinate_system = .MODEL_1.ground
!
marker modify &
marker_name = .MODEL_1.Plataforma_Movel.MARKER_46&
location = &
(LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C3))&
relative_to = .MODEL_1.Plataforma_Movel
!
defaults coordinate_system &
default_coordinate_system = .MODEL_1.ground
!
marker modify &
```



```

marker_name = .MODEL_1.Plataforma_Movel.MARKER_48 &
location = &
    (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C4)) &
relative_to = .MODEL_1.Plataforma_Movel
!

defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!

marker modify &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_50 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C5))&
    relative_to = .MODEL_1.Plataforma_Movel
!

defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!

marker modify &
    marker_name = .MODEL_1.Plataforma_Movel.MARKER_64 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.TCP))&
    relative_to = .MODEL_1.Plataforma_Movel
!

defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground

```

```
!  
geometry modify shape cylinder &  
    cylinder_name = .MODEL_1.Plataforma_Movel.  
    Plataforma&  
        length = (1.0cm) &  
        radius = (0.29173m)  
!  
geometry modify shape link &  
    link_name = .MODEL_1.Plataforma_Movel.LINK_33 &  
    width = (4.0cm) &  
    depth = (2.0cm)  
!  
geometry modify shape link &  
    link_name = .MODEL_1.Plataforma_Movel.LINK_34 &  
    width = (4.0cm) &  
    depth = (2.0cm)  
!  
geometry modify shape link &  
    link_name = .MODEL_1.Plataforma_Movel.LINK_35 &  
    width = (4.0cm) &  
    depth = (2.0cm)  
!  
marker modify &  
    marker_name = .MODEL_1.Crank2.MARKER_15 &  
    location = &
```

```

        (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.A2)) &
orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A2,
.MODEL_1.ground.B2,"Z"))&
relative_to = .MODEL_1.Crank2
!
defaults coordinate_system &
default_coordinate_system = .MODEL_1.ground
!
marker modify &
marker_name = .MODEL_1.Crank2.MARKER_29 &
location = &
        (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.A2)) &
orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A2,
.MODEL_1.ground.A2_2,"Z"))&
relative_to = .MODEL_1.Crank2
!
defaults coordinate_system &
default_coordinate_system = .MODEL_1.ground
!
marker modify &
marker_name = .MODEL_1.Crank2.MARKER_55 &
location = &
        (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.B2)) &

```

```

    relative_to = .MODEL_1.Crank2
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
geometry modify shape cylinder &
    cylinder_name = .MODEL_1.Crank2.CYLINDER_36 &
    length = (0.1939929896meter) &
    radius = (1.5cm)
!
marker modify &
    marker_name = .MODEL_1.limb2.MARKER_16 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},.MODEL_1.ground.B2))&
    orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.B2,
.MODEL_1.ground.C2,"Z"))&
    relative_to = .MODEL_1.limb2
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.limb2.MARKER_43 &
    location = &

```

```

        (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.C2))&
relative_to = .MODEL_1.limb2
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.limb2.MARKER_56 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.B2))&
relative_to = .MODEL_1.limb2
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
geometry modify shape cylinder &
    cylinder_name = .MODEL_1.limb2.CYLINDER_37 &
    length = (0.6500031767meter) &
    radius = (1.5cm)
!
marker modify &
    marker_name = .MODEL_1.Crank1.MARKER_17 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.A1)) &
orientation = &

```

```

(ORI_ALONG_AXIS(.MODEL_1.ground.A1,
.MODEL_1.ground.B1,"Z"))&
    relative_to = .MODEL_1.Crank1
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.Crank1.MARKER_31 &
    location = &
    (LOC_RELATIVE_TO({0, 0, 0}, .MODEL_1.ground.A1)) &
    orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A1,
.MODEL_1.ground.A1_2,"Z"))&
    relative_to = .MODEL_1.Crank1
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.Crank1.MARKER_53 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
        .MODEL_1.ground.B1))&
    relative_to = .MODEL_1.Crank1

```

```

!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!

geometry modify shape cylinder &
    cylinder_name = .MODEL_1.Crank1.CYLINDER_38 &
    length = (0.1939998064meter) &
    radius = (1.5cm)
!

marker modify &
    marker_name = .MODEL_1.limb1.MARKER_18 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
            .MODEL_1.ground.B1)) &
    orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.B1,
.MODEL_1.ground.C1,"Z"))&
    relative_to = .MODEL_1.limb1
!

defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!

marker modify &
    marker_name = .MODEL_1.limb1.MARKER_41 &
    location = &

```

```
(LOC_RELATIVE_TO({0, 0, 0},
    .MODEL_1.ground.C1)) &
relative_to = .MODEL_1.limb1
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.limb1.MARKER_54 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
            .MODEL_1.ground.B1)) &
    relative_to = .MODEL_1.limb1
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
geometry modify shape cylinder &
    cylinder_name = .MODEL_1.limb1.CYLINDER_39 &
    length = (0.6500000857meter) &
    radius = (1.5cm)
!
marker modify &
    marker_name = .MODEL_1.Crank6.MARKER_19 &
    location = &
```



```

        (LOC_RELATIVE_TO({0, 0, 0},
        .MODEL_1.ground.A6)) &
orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A6,
.MODEL_1.ground.B6,"Z"))&
relative_to = .MODEL_1.Crank6
!
defaults coordinate_system &
default_coordinate_system = .MODEL_1.ground
!
marker modify &
marker_name = .MODEL_1.Crank6.MARKER_51 &
location = &
        (LOC_RELATIVE_TO({0, 0, 0},
        .MODEL_1.ground.B6)) &
relative_to = .MODEL_1.Crank6
!
defaults coordinate_system &
default_coordinate_system = .MODEL_1.ground
!
geometry modify shape cylinder &
cylinder_name = .MODEL_1.Crank6.CYLINDER_40 &
length = (0.1939904637meter) &
radius = (1.5cm)
!

```

```
marker modify &

    marker_name = .MODEL_1.Crank6.MARKER_79 &

    location = &

        (LOC_RELATIVE_TO({0, 0, 0},

            .MODEL_1.ground.A6))&

    orientation = &

(ORI_ALONG_AXIS(.MODEL_1.ground.A6,

.MODEL_1.ground.A6_2,"Z"))&

    relative_to = .MODEL_1.Crank6

!

defaults coordinate_system &

    default_coordinate_system = .MODEL_1.ground

!

marker modify &

    marker_name = .MODEL_1.limb6.MARKER_20 &

    location = &

        (LOC_RELATIVE_TO({0, 0, 0},

            .MODEL_1.ground.B6))&

    orientation = &

(ORI_ALONG_AXIS(.MODEL_1.ground.B6,

.MODEL_1.ground.C6,"Z"))&

    relative_to = .MODEL_1.limb6

!

defaults coordinate_system &

    default_coordinate_system = .MODEL_1.ground
```

```
!  
marker modify &  
    marker_name = .MODEL_1.limb6.MARKER_39 &  
    location = &  
(LOC_RELATIVE_TO({0, 0, 0},  
.MODEL_1.ground.C6)) &  
    relative_to = .MODEL_1.limb6  
!  
defaults coordinate_system &  
    default_coordinate_system = .MODEL_1.ground  
!  
marker modify &  
    marker_name = .MODEL_1.limb6.MARKER_52 &  
    location = &  
    (LOC_RELATIVE_TO({0, 0, 0},  
    .MODEL_1.ground.B6))&  
    relative_to = .MODEL_1.limb6  
!  
defaults coordinate_system &  
    default_coordinate_system = .MODEL_1.ground  
!  
geometry modify shape cylinder &  
    cylinder_name = .MODEL_1.limb6.CYLINDER_41 &  
    length = (0.6500005945meter) &  
    radius = (1.5cm)
```

```
!  
marker modify &  
    marker_name = .MODEL_1.Crank5.MARKER_21 &  
    location = &  
        (LOC_RELATIVE_TO({0, 0, 0},  
        .MODEL_1.ground.A5)) &  
    orientation = &  
(ORI_ALONG_AXIS(.MODEL_1.ground.A5,  
.MODEL_1.ground.B5,"Z")) &  
    relative_to = .MODEL_1.Crank5  
!  
defaults coordinate_system &  
    default_coordinate_system = .MODEL_1.ground  
!  
marker modify &  
    marker_name = .MODEL_1.Crank5.MARKER_35 &  
    location = &  
        (LOC_RELATIVE_TO({0, 0, 0},  
        .MODEL_1.ground.A5)) &  
    orientation = &  
(ORI_ALONG_AXIS(.MODEL_1.ground.A5,  
.MODEL_1.ground.A5_2,"Z")) &  
    relative_to = .MODEL_1.Crank5  
!  
defaults coordinate_system &
```

```

    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.Crank5.MARKER_61 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
        .MODEL_1.ground.B5)) &
    relative_to = .MODEL_1.Crank5
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
geometry modify shape cylinder &
    cylinder_name = .MODEL_1.Crank5.CYLINDER_42 &
    length = (0.1939989023meter) &
    radius = (1.5cm)
!
marker modify &
    marker_name = .MODEL_1.limb5.MARKER_22 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
        .MODEL_1.ground.B5)) &
    orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.B5,
.MODEL_1.ground.C5,"Z")) &

```

```
relative_to = .MODEL_1.limb5
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.limb5.MARKER_49 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
            .MODEL_1.ground.C5)) &
    relative_to = .MODEL_1.limb5
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.limb5.MARKER_62 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
            .MODEL_1.ground.B5)) &
    relative_to = .MODEL_1.limb5
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
```

```

geometry modify shape cylinder &
    cylinder_name = .MODEL_1.limb5.CYLINDER_43 &
    length = (0.6500021103meter) &
    radius = (1.5cm)
!
marker modify &
    marker_name = .MODEL_1.Crank4.MARKER_23 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
        .MODEL_1.ground.A4)) &
    orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A4,
.MODEL_1.ground.B4,"Z")) &
    relative_to = .MODEL_1.Crank4
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.Crank4.MARKER_37 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
        .MODEL_1.ground.A4)) &
    orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A4,

```

```
.MODEL_1.ground.A4_2,"Z")) &
    relative_to = .MODEL_1.Crank4
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.Crank4.MARKER_60 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
            .MODEL_1.ground.B4)) &
    relative_to = .MODEL_1.Crank4
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
geometry modify shape cylinder &
    cylinder_name = .MODEL_1.Crank4.CYLINDER_44 &
    length = (0.1940012969meter) &
    radius = (1.5cm)
!
marker modify &
    marker_name = .MODEL_1.limb4.MARKER_24 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
```



```

        .MODEL_1.ground.B4))    &
orientation =    &
(ORI_ALONG_AXIS(.MODEL_1.ground.B4,
.MODEL_1.ground.C4,"Z"))    &
relative_to = .MODEL_1.limb4
!
defaults coordinate_system    &
default_coordinate_system = .MODEL_1.ground
!
marker modify    &
marker_name = .MODEL_1.limb4.MARKER_47    &
location =    &
(LOC_RELATIVE_TO({0, 0, 0},
.MODEL_1.ground.C4))    &
relative_to = .MODEL_1.limb4
!
defaults coordinate_system    &
default_coordinate_system = .MODEL_1.ground
!
marker modify    &
marker_name = .MODEL_1.limb4.MARKER_59    &
location =    &
(LOC_RELATIVE_TO({0, 0, 0},
.MODEL_1.ground.B4))    &
relative_to = .MODEL_1.limb4

```

```
!  
defaults coordinate_system &  
    default_coordinate_system = .MODEL_1.ground  
!  
geometry modify shape cylinder &  
    cylinder_name = .MODEL_1.limb4.CYLINDER_45 &  
    length = (0.6500012678meter) &  
    radius = (1.5cm)  
!  
marker modify &  
    marker_name = .MODEL_1.Crank3.MARKER_25 &  
    location = &  
        (LOC_RELATIVE_TO({0, 0, 0},  
            .MODEL_1.ground.A3)) &  
    orientation = &  
(ORI_ALONG_AXIS(.MODEL_1.ground.A3,  
.MODEL_1.ground.B3,"Z"))&  
    relative_to = .MODEL_1.Crank3  
!  
defaults coordinate_system &  
    default_coordinate_system = .MODEL_1.ground  
!  
marker modify &  
    marker_name = .MODEL_1.Crank3.MARKER_27 &  
    location = &
```

```

        (LOC_RELATIVE_TO({0, 0, 0},
            .MODEL_1.ground.A3)) &
orientation = &
(ORI_ALONG_AXIS(.MODEL_1.ground.A3,
.MODEL_1.ground.A3_2,"Z"))&
relative_to = .MODEL_1.Crank3
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
marker modify &
    marker_name = .MODEL_1.Crank3.MARKER_57 &
    location = &
        (LOC_RELATIVE_TO({0, 0, 0},
            .MODEL_1.ground.B3)) &
    relative_to = .MODEL_1.Crank3
!
defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!
geometry modify shape cylinder &
    cylinder_name = .MODEL_1.Crank3.CYLINDER_46 &
    length = (0.1939977582meter) &
    radius = (1.5cm)
!

```

```
marker modify &

    marker_name = .MODEL_1.limb3.MARKER_26 &

    location = &

        (LOC_RELATIVE_TO({0, 0, 0},

            .MODEL_1.ground.B3)) &

    orientation = &

(ORI_ALONG_AXIS(.MODEL_1.ground.B3,

.MODEL_1.ground.C3,"Z"))&

    relative_to = .MODEL_1.limb3

!

defaults coordinate_system &

    default_coordinate_system = .MODEL_1.ground

!

marker modify &

    marker_name = .MODEL_1.limb3.MARKER_45 &

    location = &

        (LOC_RELATIVE_TO({0, 0, 0},

            .MODEL_1.ground.C3)) &

    relative_to = .MODEL_1.limb3

!

defaults coordinate_system &

    default_coordinate_system = .MODEL_1.ground

!

marker modify &

    marker_name = .MODEL_1.limb3.MARKER_58 &
```

```
location =      &
    (LOC_RELATIVE_TO({0, 0, 0},
        .MODEL_1.ground.B3))  &
relative_to = .MODEL_1.limb3
!

defaults coordinate_system &
    default_coordinate_system = .MODEL_1.ground
!

geometry modify shape cylinder &
    cylinder_name = .MODEL_1.limb3.CYLINDER_47  &
    length = (0.6500030919meter)  &
    radius = (1.5cm)
!

model display &
    model_name = MODEL_1
```

9.2 Positions Points File

```
point create point_name = .MODEL_1.ground.TCP
location = 0.00000, 0.00000, -0.45000
point create point_name = .MODEL_1.ground.A1
location = 0.63689, 0.19756, 0.00000
point create point_name = .MODEL_1.ground.A2
location = -0.14735, 0.65034, 0.00000
point create point_name = .MODEL_1.ground.A3
location = -0.48953, 0.45278, 0.00000
point create point_name = .MODEL_1.ground.A4
location = -0.48953, -0.45278, 0.00000
point create point_name = .MODEL_1.ground.A5
location = -0.14735, -0.65034, 0.00000
point create point_name = .MODEL_1.ground.A6
location = 0.63689, -0.19756, 0.00000
point create point_name = .MODEL_1.ground.A1_2
location = 0.46368, 0.09756, 0.00000
point create point_name = .MODEL_1.ground.A2_2
location = -0.14735, 0.85034, 0.00000
point create point_name = .MODEL_1.ground.A3_2
location = -0.31633, 0.35278, 0.00000
point create point_name = .MODEL_1.ground.A4_2
location = -0.66274, -0.55278, 0.00000
```

```
point create point_name = .MODEL_1.ground.A5_2
location = -0.14735, -0.45034, 0.00000
point create point_name = .MODEL_1.ground.A6_2
location = 0.81009, -0.29756, 0.00000
point create point_name = .MODEL_1.ground.B1
location = 0.56640, 0.31964, 0.13328
point create point_name = .MODEL_1.ground.B2
location = -0.31125, 0.65034, -0.10379
point create point_name = .MODEL_1.ground.B3
location = -0.56001, 0.33070, 0.13328
point create point_name = .MODEL_1.ground.B4
location = -0.40758, -0.59472, -0.10379
point create point_name = .MODEL_1.ground.B5
location = -0.00639, -0.65034, 0.13328
point create point_name = .MODEL_1.ground.B6
location = 0.71883, -0.05561, -0.10379
point create point_name = .MODEL_1.ground.C1
location = 0.29173, 0.40235, -0.45000
point create point_name = .MODEL_1.ground.C2
location = 0.20258, 0.45382, -0.45000
point create point_name = .MODEL_1.ground.C3
location = -0.49431, 0.05147, -0.45000
point create point_name = .MODEL_1.ground.C4
location = -0.49431, -0.05147, -0.45000
point create point_name = .MODEL_1.ground.C5
```

```
location = 0.20258, -0.45382, -0.45000
```

```
point create point_name = .MODEL_1.ground.C6
```

```
location = 0.29173, -0.40235, -0.45000
```