DESIGN OF A PARALLEL SCHÖNFLIES MOTION GENERATOR

Ravi Singaram Sergio Uribe

January 22, 2020

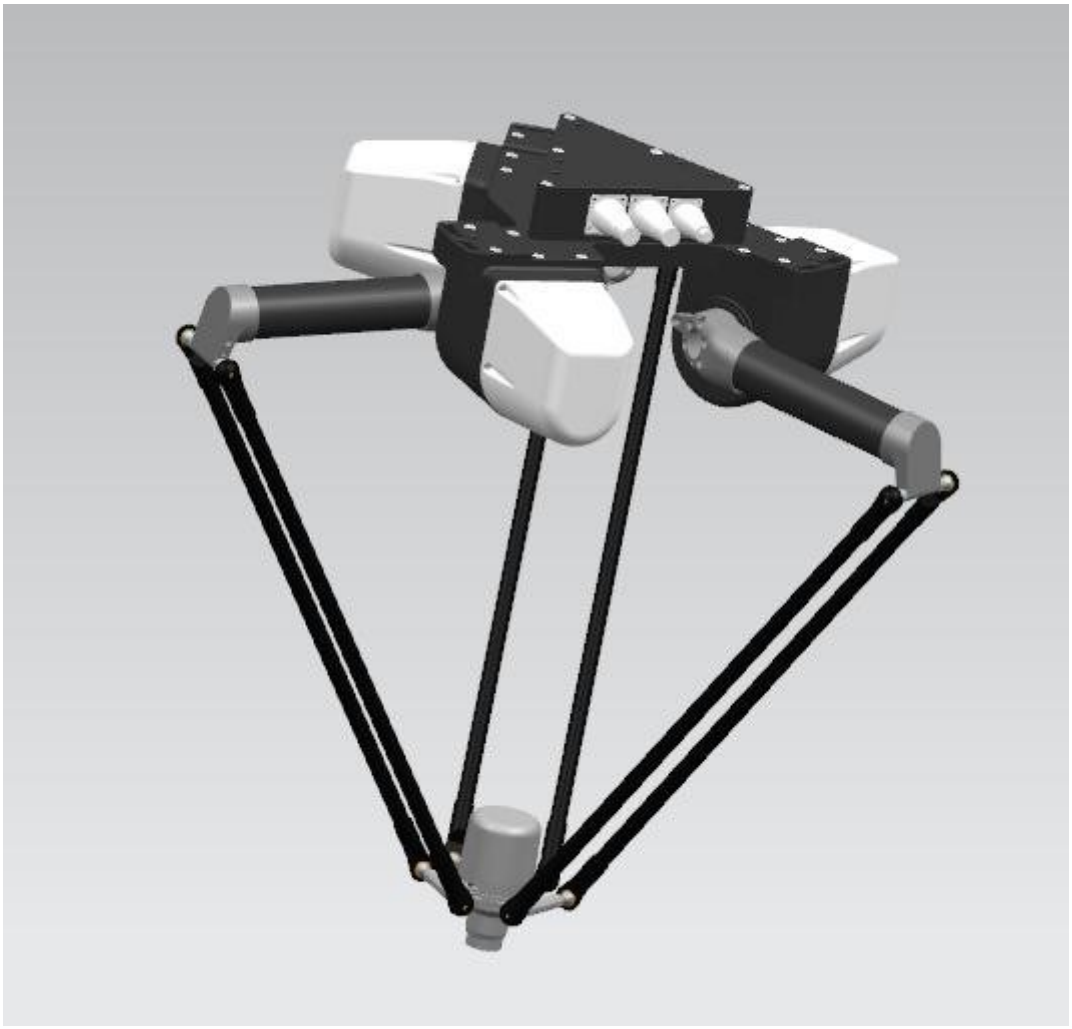


Figure 1: Schönflies Motion

Summary

List of Figures

List of Tables

1 Introduction

The subject of Project 3 is about the design of a parallel Schönflies Motion Generator (SMG). The parallel Schönflies motion generator should be designed for pick-and-place operations based on the path defined to test the performance of SCARA robots. Moreover, one of the objectives of the project is to realize a parallel system that outperforms both serial SCARA systems and the parallel robots currently available for Schönflies motion generation.

2 Specifications to be fulfilled

1) The robot must be capable of producing a test cycle that is commonly accepted for SCARA systems. The test cycle consists of: – 25-mm vertical displacement up – 300-mm horizontal displacement with a concomitant 90° turn – 25-mm vertical displacement down – 25-mm vertical displacement up – 300-mm horizontal displacement with a reversed 90° turn – 25-mm vertical displacement down.

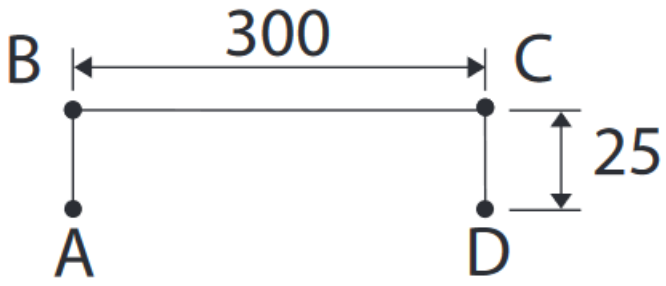


Figure 2: Path adopted for the manipulator design

2) The system should be capable of attaining a configuration where kinetostatic robust- ness is achieved. As a matter of fact, from kinetostatics, the SMG is maximally ac- curate and robust when its normalized forward and inverse Jacobians are maximally invertible, i.e., when their condition number κ is as small as possible, i.e., $\kappa < 10$.

3) The mechanism should not be too bulky.

3 Structure of the report

The structure of the report has been defined by the steps and the process followed in the project.
1) Design of an non-parameterized manipulator is presented.
2) Twist and wrench analysis. It gives the motion of the moving platform.
3) Inverse Geometric Model of the Manipulator.
4) Computation of the Jacobian matrix and the Condition Number.
5) Optimization problem.

4 Non-parameterized Manipulator: Schönflies motion generator

Having the Schönflies motion is the same as saying that the moving platform of the manipulator must have the 3 possible translations and rotation along the vertical Z axis. It can be obtained with a 4-RUU manipulator but also with some other manipulators, for example, a 4-ŔŔŔŔŔ manipulator.
At the beginning, we started the project by doing the design of this last manipulator. The following configuration was designed:

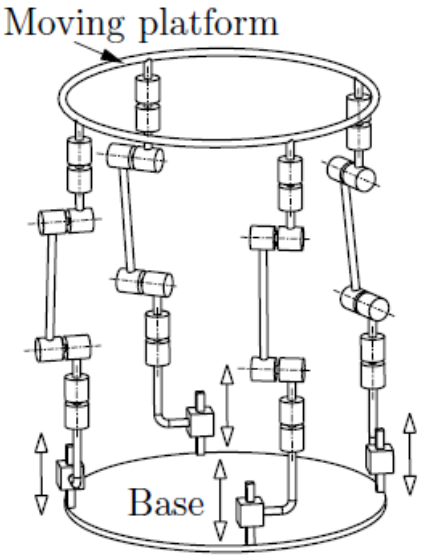
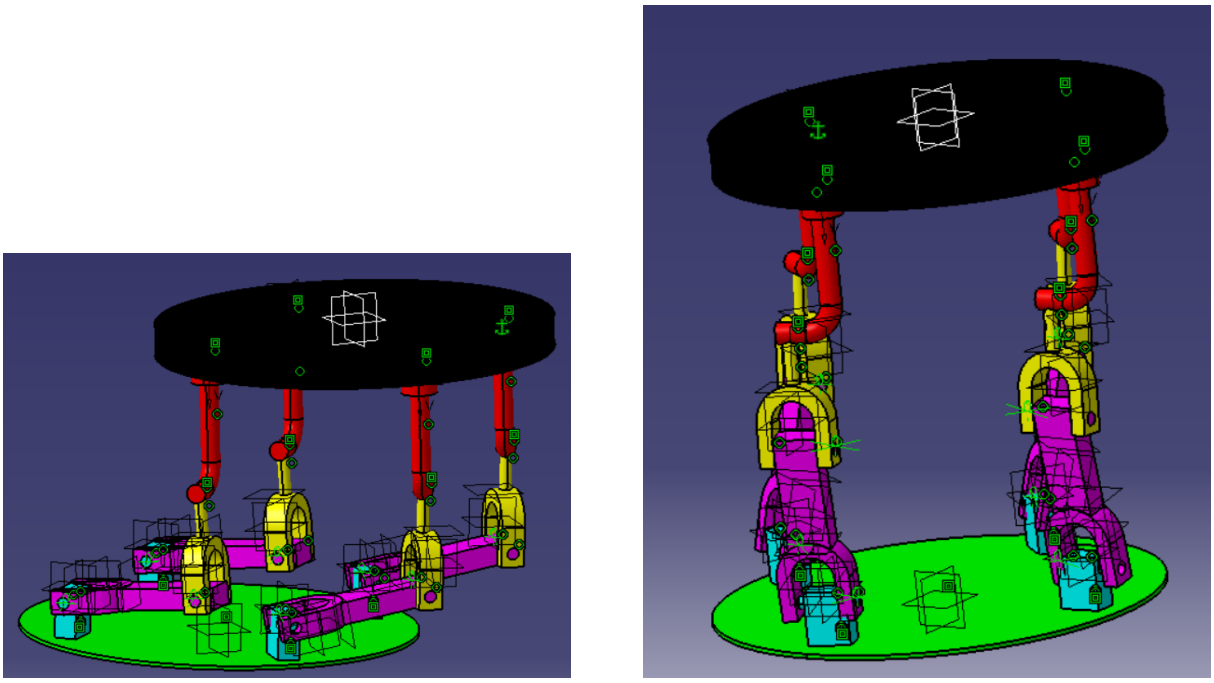


Figure 3: 4-ŔŔŔŔŔ Manipulator Configuration.

Some problems were faced when designing it: taking a look at Figure 4, in the beginning, the last revolute joint was designed attaching directly the blue part to the moving platform. However, as they are attached in 4 points, the rotation along Z was not possible, and only 2 degrees of freedom were possible. Consequently, we solved this issue designing another part allowing the rotation in Z, but some other problems appear.
The design of the different parts were also problematic. Focusing on the red part, its height is too big, while its length in the XY plane should be larger so as to have a bigger workspace. Besides, with this design, the mechanism is bulkier, just the opposite of what is desired.
As some problems occurred when simulating the mechanism, finally the decision of designing an optimal mechanism was made. We took a real 4-RUU physically and we reproduced its design in Catia (Figure 5).



(a) Minimum height. (b) Maximum height.

Figure 4: 4-ŔŔR̀R̀ Manipulator

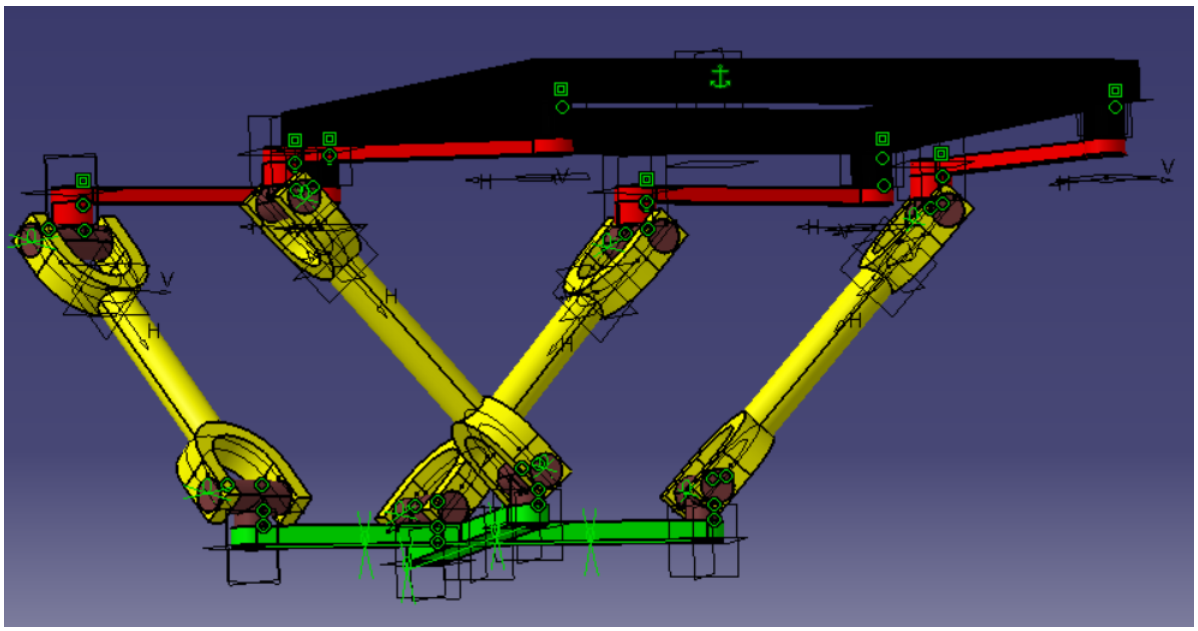


Figure 5: 4-RUU Manipulator.

This manipulator works fine, having the moving platform the Schönflies motion, in other words, the 3 possible translations and a rotation along Z. At first, it was designed without fulfilling any of the specifications of section 2, just checking that the motion was the desired one. In this part of the report, a detailed view of the revolute and universal joints is given, since we need to understand how they work for the twist and wrench analysis section, in which this joints play an important role.
Revolute joints provide single-axis rotation. Universal joints, however, are a coupling between two rotating shafts allowing freedom of angular movement in all directions. It can be seen as 2 revolute joints.

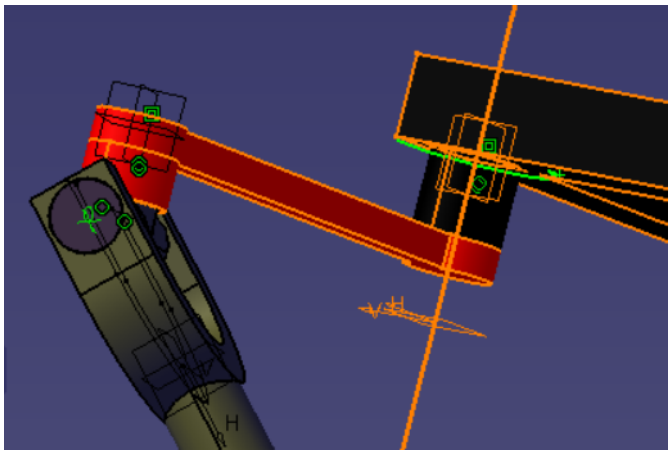
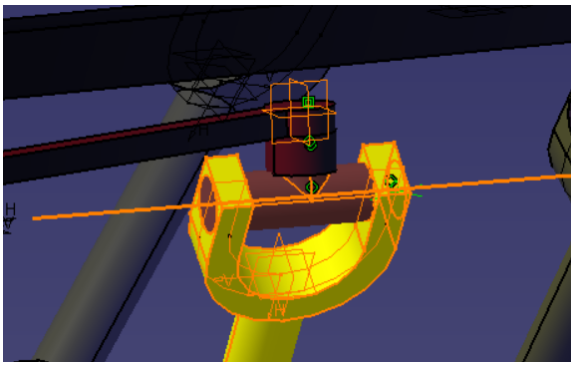
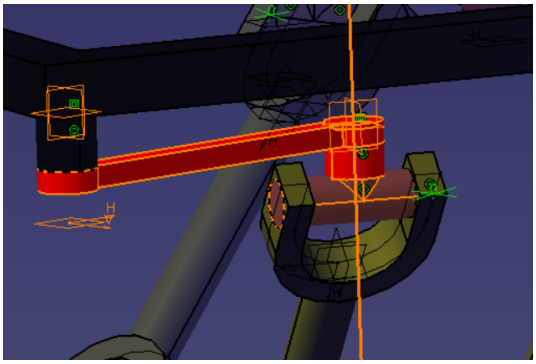


Figure 6: Revolute Joint.



(a) First revolute joint.



(b) Second revolute joint.

Figure 7: Universal joint.

5 Twist and wrench analysis

This analysis is useful for the knowledge of the motion of the moving platform. Besides it will give us the Direct and Inverse Jacobian matrices which will allow us to compute the Jacobian matrix.
From the screw theory we know that a zero pitch screw (Plücker coordinate vector of a finite line) is defined as:

$$\hat{\$}_0 = \begin{pmatrix} s \\ rxs \end{pmatrix} \tag{1}$$

while an infinite pitch screw (Plücker coordinate vector of an infinite line) is defined as:

$$\hat{\$}_\infty = \begin{pmatrix} 0 \\ s \end{pmatrix} \tag{2}$$

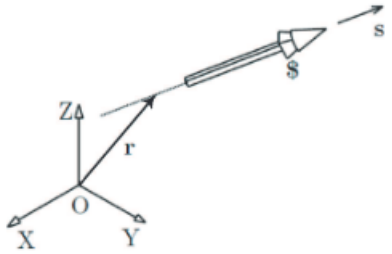


Figure 8: Screw Theory: r and s vectors.

We can follow these steps for the twist and wrench analysis:

Step 1: Limb twist system.
The twist system τ of the parallel kinematic chain is the intersection of the twist system τ_i of all its serial chains.

$$\tau_i = span(\hat{\varepsilon}_{0i1}, \hat{\varepsilon}_{0i2}, \hat{\varepsilon}_{0i3}, \hat{\varepsilon}_{0i4}, \hat{\varepsilon}_{0i5})$$

$$\text{with } \hat{\varepsilon}_{0i1} = \begin{pmatrix} k \\ r_{Ai}xk \end{pmatrix}, \hat{\varepsilon}_{0i2} = \begin{pmatrix} k \\ r_{Bi}xk \end{pmatrix}, \hat{\varepsilon}_{0i3} = \begin{pmatrix} m_i \\ r_{Bi}xm_i \end{pmatrix}, \hat{\varepsilon}_{0i4} = \begin{pmatrix} m_i \\ r_{Ci}xm_i \end{pmatrix}, \hat{\varepsilon}_{0i1} = \begin{pmatrix} k \\ r_{Ci}xk \end{pmatrix}.$$

Step 2: Limb constraint wrench system.
Its wrench system ω is the linear combination of the wrench systems ω_i of all its serial chains. The constraint wrench system includes wrenches that are reciprocal to all its twists. In this case, an infinite pitch wrench is reciprocal to all twists.

$$\hat{\tau}_{\infty i} = \begin{pmatrix} 0_3 \\ kxm_i \end{pmatrix}$$

$$\omega_i = span(\hat{\tau}_{\infty i})$$

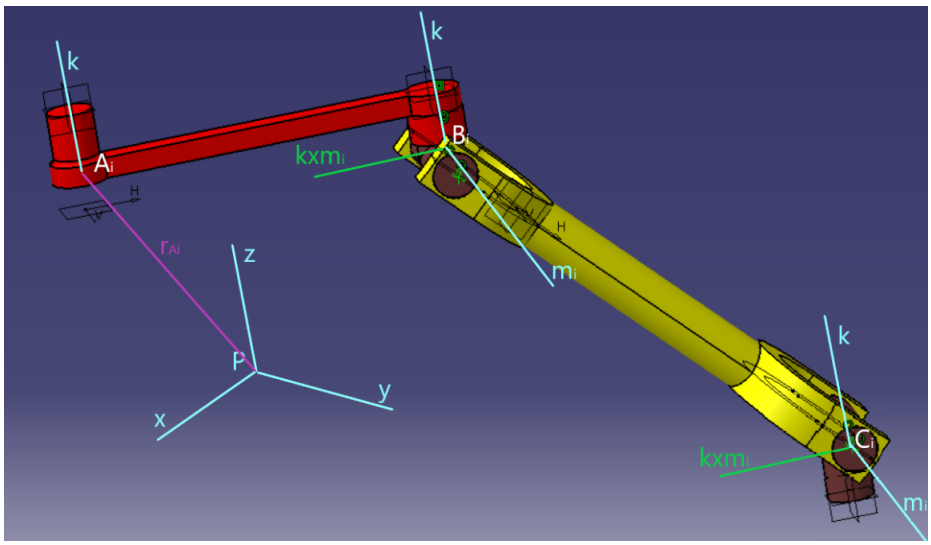


Figure 9: Twist and wrench system of the limb i.

Step 3: Constraint wrench system.
The constraint wrench system for the whole mechanism is defined as:

$$W_c = \sum_{k=1}^{4} \omega_i = span(\hat{\tau}_{\infty 1}, \hat{\tau}_{\infty 2}, \hat{\tau}_{\infty 3}, \hat{\tau}_{\infty 4})$$

W_c is a 2-system. The 4 constraint wrenches are in XY plane, so they are linearly dependent 2 to 2.

Step 4: Twist system of the moving-platform.

$$\tau = span(\hat{\varepsilon}_{\infty x}, \hat{\varepsilon}_{\infty y}, \hat{\varepsilon}_{\infty z}, \hat{\varepsilon}_{0k})$$

The twist system of the moving platform is defined as 3 translations in X, Y and Z axis and a rotation along Z axis. This is exactly what we want to achieve, corresponding to the Schönflies motion.

Step 5: Actuation wrench system.
The actuation wrench system includes the additional wrenches obtained by locking actuators. In a general configuration, when we block actuators, the robot is fully constrained, cannot provide any motion.

$$\hat{\tau}_{0a} = \begin{pmatrix} f_i \\ r_{Ci} x f_i \end{pmatrix}$$

$$W_a = span(\hat{\tau}_{0a1}, \hat{\tau}_{0a2}, \hat{\tau}_{0a3}, \hat{\tau}_{0a4})$$

W_a is a 4-system.

Step 6: Global wrench system.
Constraint and actuation wrenches form a 6-wrench system called global wrench system.

$$W_G = W_a \oplus W_c = span(\hat{\tau}_{0a1}, \hat{\tau}_{0a2}, \hat{\tau}_{0a3}, \hat{\tau}_{0a4}, \hat{\tau}_{\infty 1}, \hat{\tau}_{\infty 2}, \hat{\tau}_{\infty 3}, \hat{\tau}_{\infty 4})$$

W_G is a 6-system.
Finally, we can compute the direct and inverse Jacobian matrices, fulfilling the following equation:

$$At = B\dot{\theta} \tag{3}$$

being

$$t = \begin{pmatrix} w \\ \dot{p} \end{pmatrix}$$

$$\dot{\theta} = \begin{pmatrix} \dot{\theta}_{11} \\ \dot{\theta}_{21} \\ \dot{\theta}_{31} \\ \dot{\theta}_{41} \end{pmatrix} \tag{4}$$

Consequently, we obtain the matrices A and B as:

$$A = \begin{bmatrix} (r_{C1} x f_1)^T & f_1^T \\ (r_{C2} x f_2)^T & f_2^T \\ (r_{C3} x f_3)^T & f_3^T \\ (r_{C4} x f_4)^T & f_4^T \\ (i)^T & 0_3^T \\ (j)^T & 0_3^T \end{bmatrix} \tag{5}$$

$$B = \begin{bmatrix} (\overrightarrow{A_1 C_1} x f_1)^T k & 0 & 0 & 0 \\ 0 & (\overrightarrow{A_2 C_2} x f_2)^T k & 0 & 0 \\ 0 & 0 & (\overrightarrow{A_3 C_3} x f_3)^T k & 0 \\ 0 & 0 & 0 & (\overrightarrow{A_4 C_4} x f_4)^T k \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

These matrices will be useful in section 7.
When computing these matrices we faced 2 problems:
- Matrix A is not homogeneous. This means that the units are not all the same, as f_1 is a unitary vector and r_{Ci} is in meters. This could be solved by dividing this values by a characteristic length L, which can be approximated to the design parameter r of the moving platform. However, we will not consider this in the computation of the Jacobian. Therefore, the specification of the condition number being greater than 10 could be modified to being greater than 100.
- The definition of m_i for the constraint wrench is too complex, so what we did in the last 2 rows of the matrix A is to constraint the entire plane XY. However, there is one special case in which the constraint wrench system is not the plane XY. This happens when all m_i are parallel. This is a parallel singularity in which the rotation along m_i is allowed. The following assumption is made: this singularity is neglected and assumed not to happen.

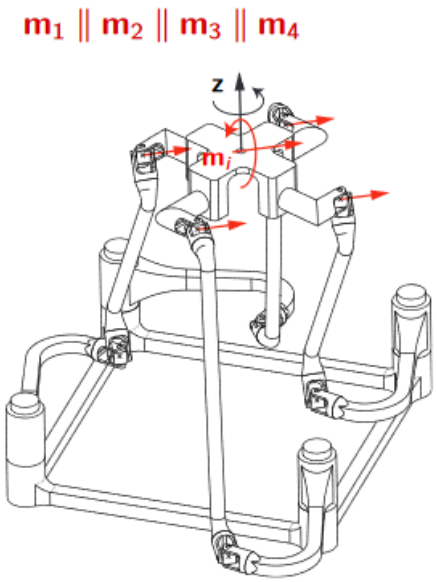


Figure 10: Constraint singular configuration.

6 Inverse Geometric Model(IGM) of the 4-RUU Manipulator

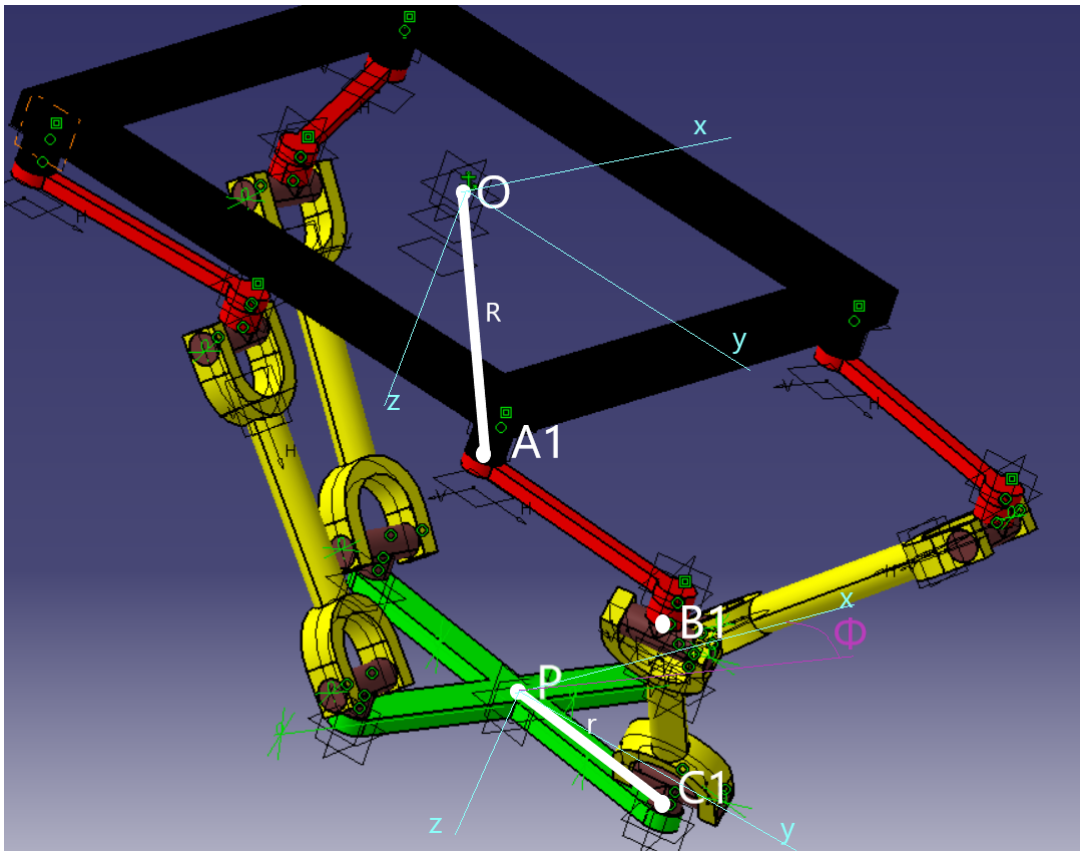


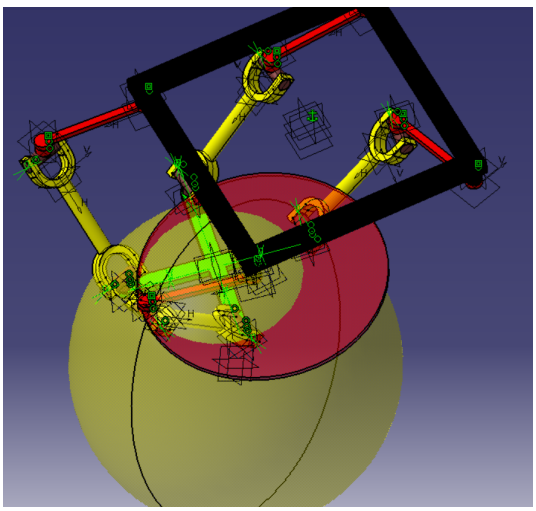
Figure 11: Adopted notation.

For computing the Inverse Geometric Model, many things have to be considered. First of all, it is important to localise the intersecting points between axis denoted as O, A1, B1 and C1, the index 1 refering to limb 1.Note the following design variables of the manipulator:
- L1: length of leg 1, distance between A_i and B_i.
- L2: length of leg 2, distance between B_i and C_i.
- R: distance between O(center of the fixed frame) and A_i, being i a value between 1 and 4(the manipulator has 4 limbs).
- r: distance between the point P and C_i.
The point P is the center of the moving platform. This is the point where the end-effector of the manipulator remains. The inverse geometric model (IGM) provides the joint variables corresponding to a given location of the end-effector. Therefore, the cartesian coordinates of point P and the angle of the moving platform Φ are known: $\{x_P, y_P, z_P, \phi\}$.
The knowledge of the end-effector's position allows us to define the coordinates of point C_i, as the value of r is known. It is a design parameter of the manipulator.

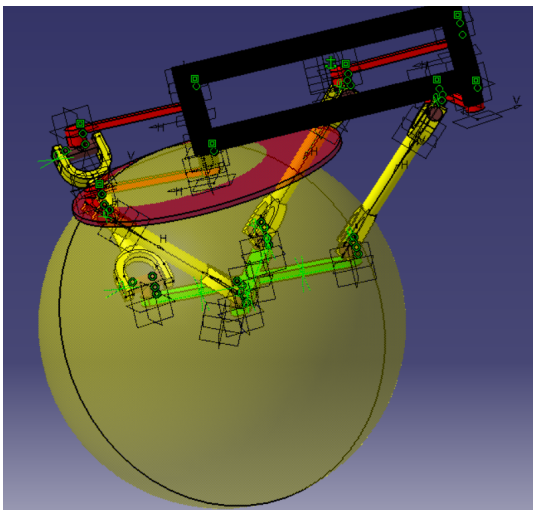
$$\{x_P \pm r * cos(\phi), y_P \pm r * sin(\phi), z_P\} \text{ or } \{x_P \pm r * sin(\phi), y_P \pm r * cos(\phi), z_P\}.$$

The fixed frame is at point O. Its coordinates are chosen to be in $\{0,\ 0,\ 0\}$. Thus, the coordinates of A_i can also be defined using the design parameter R:

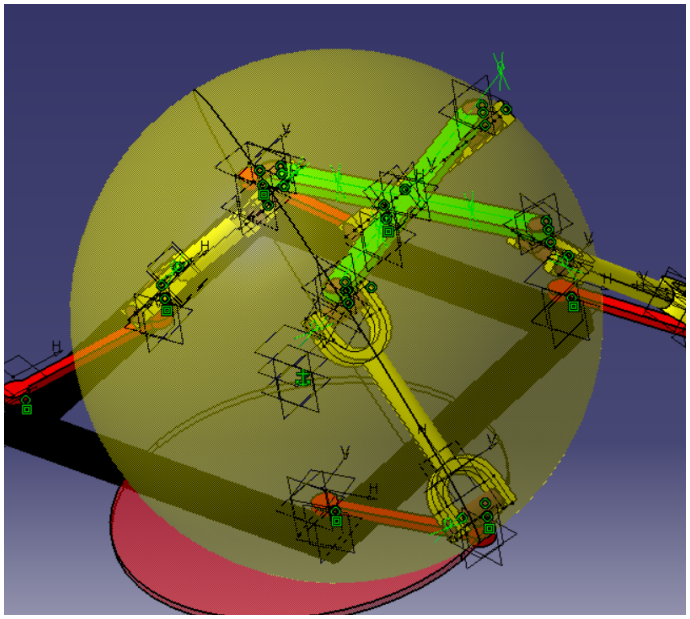
$$\{\pm R * cos(45^o), \pm R * sin(45^o), 0\}$$



(a) Top view.



(b) Side view.



(c) Below view.

Figure 12: Intersection between the sphere and the circle.

The coordinates of points B_i are unknown. This points are the intersection between the leg A_iB_i and the leg B_iC_i. The workspace of the first leg is a circle in the plane XY, which can be defined as:

$$(X - X_{A_i})^2 + (Y - Y_{A_i})^2 = L1^2 \tag{7}$$

being $L1$ the length of the leg and the radius of the circle.

The workspace of the second leg is a sphere, as this leg has 2 possibilities of rotation. This sphere is given by:

$$(X - X_{C_i})^2 + (Y - Y_{C_i})^2 + (Z - Z_{C_i})^2 = L2^2 \tag{8}$$

being $L2$ the length of the leg and the radius of the sphere.
See Figure 12.

The sphere can be projected in the XY plane, so that the intersection of 2 circles can be done(without the component in Z axis). However, the radius of the circle has to be modified. The projection of $L2$ degenerates in n, as it is shown in Figure 13.

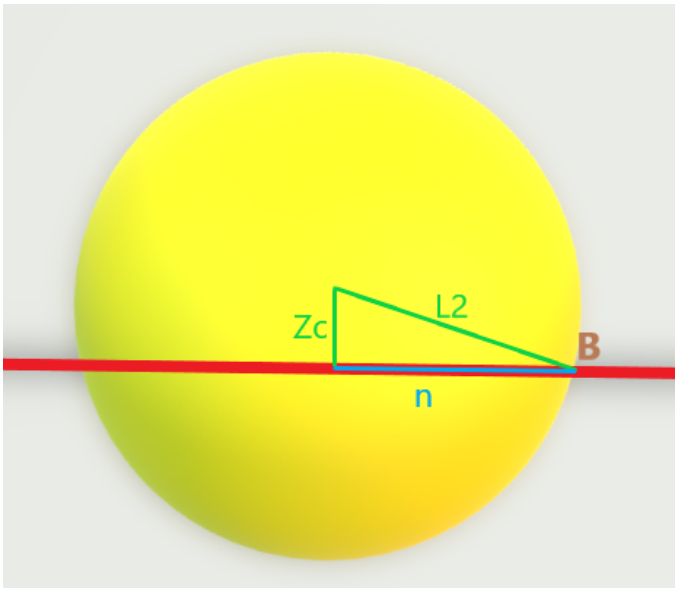


Figure 13: Projection of L2: n.

The equation (2) can be expressed now as:

$$(X - X_{C_i})^2 + (Y - Y_{C_i})^2 = n^2 \tag{9}$$

At this point, there are many different ways to do the intersection between the 2 circles. 3 ways with different purposes have been made in MATLAB. The scripts are attached to this report.

1) Using the function **syms** of MATLAB so that to obtain the IGM in a generic form, in terms of the design variables of the manipulator: $L1, L2, R$ and r. Then, the function **solve** has been used to obtain the intersection points. This approach is useful for the computation afterwards of the Jacobian in terms of the design variables. However, for this project is not really useful because the matrices are huge.

2) Using the function **circcirc** of MATLAB: the intersection of circles is made in the Cartesian plane, giving the coordinates of the 2 points of intersection. This approach has been done to obtain the coordinates of points B_i with values, so that to check that the IGM works in our manipulator with specific values for design parameters.

3) Geometrically compute the required angles. The IGM seeks to compute the actuated joints θ_i. Taking a look to Figure 14, the angle α can be defined using the Law of Cosine, while γ is computed with the coordinates of A_i and C_i. Finally, θ is the addition of the 2 angles.

$$\cos(\alpha) = \frac{L1^2 + (\overrightarrow{AC})^2 - n^2}{2 * L1 * \overrightarrow{AC}} \tag{10}$$

$$\tan(\gamma) = \frac{Y_{C_i} - Y_{A_i}}{X_{C_i} - X_{A_i}} \tag{11}$$

$$\theta = \alpha + \gamma \tag{12}$$

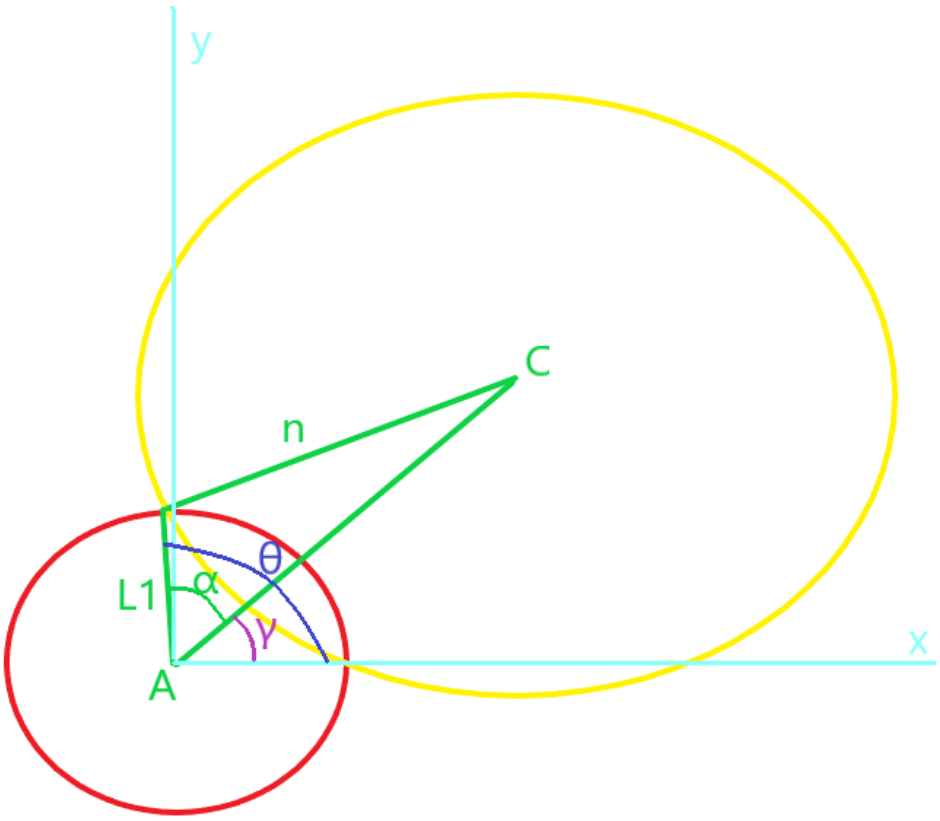


Figure 14: IGM geometrically: Angle theta.

The second technique has been used to check that the IGM works fine and that the specification of the trajectory can be fulfilled. This script gives us the values of the actuated joints for any point. When a point is not reachable, the code gives the following output: *Not possible to reach this point.*

Checking the IGM with Catia, we faced a big issue: despite we defined all the θ_i emerging from X axis, in Catia they were defined very differently. In Figure 15, the actuated joints are shown for a null value. As it can be seen, the angle θ_i seems to emerge each one from a different axis. This made really difficult to check properly that the Inverse Geometric Model was working. However, we did an analysis in MATLAB to see if the output makes sense. These have been the results for the adopted θ_i of Figure 16. Note that there are 2 possible outputs for each θ_i as we are considering as valid the 2 points of the intersection.

$L1 = 6.5, L2 = 12, R = 12, r = 6[cm]$	Θ_1	Θ_2	Θ_3	Θ_4
$X_P = 0, Y_P = 0, Z_P = 11, \phi = 0^o$	87º or 54º	-2º or 35º	-87º or -54º	2º or -35º
$X_P = 5, Y_P = 0, Z_P = 9, \phi = 30^o$	86º or -19º	-21º or -4º	7º or -26º	28º or -62º
$X_P = 5, Y_P = 7, Z_P = 9, \phi = 30^o$	Not possible to reach this point.			
$X_P = 0, Y_P = 7, Z_P = 5, \phi = 45^o$	46º or 4º	4º or 46º	5º or -71º	-71º or 5º
$X_P = 1, Y_P = 2, Z_P = 5, \phi = 120^o$	75º or -29º	28º or 48º	-82º or 68º	10º or -37º

Table 1: Values of θ_i for different coordinates and orientation of point P.

These outputs makes sense and all values are possible for those coordinates of point P. The third point checked is not reachable because a serial singularity happens in limb 1. Point P is far away from A_1 and the maximum distance the limb can reach is $L1 + L2 = 18.5$.

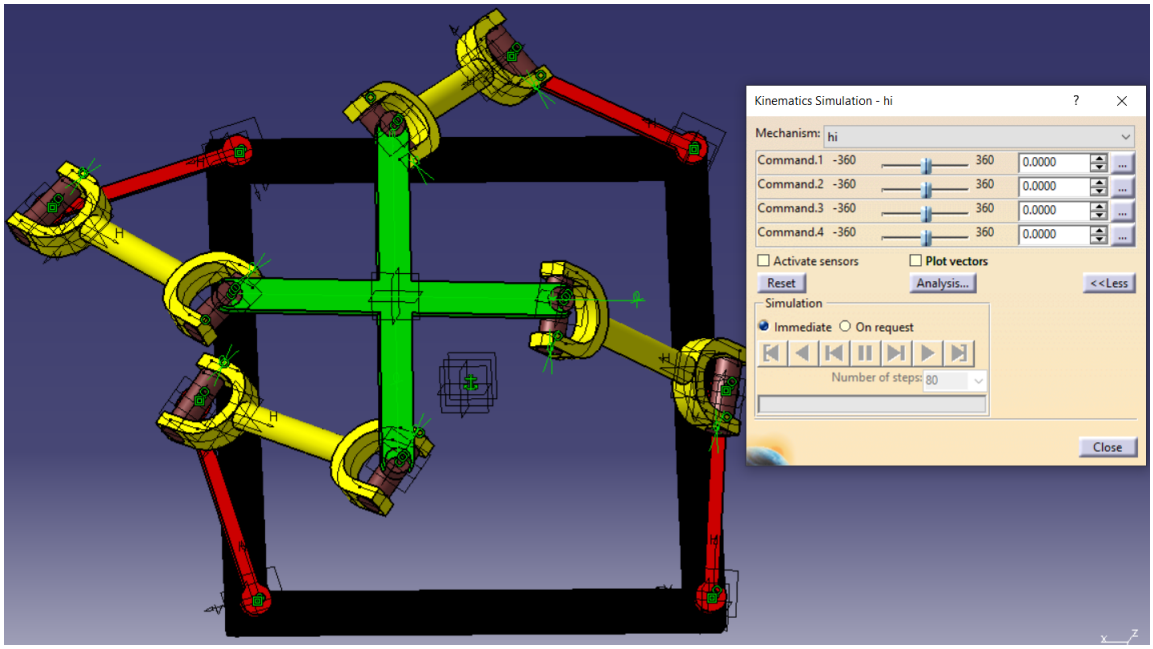


Figure 15: Manipulator's configuration for $\theta_i = 0$

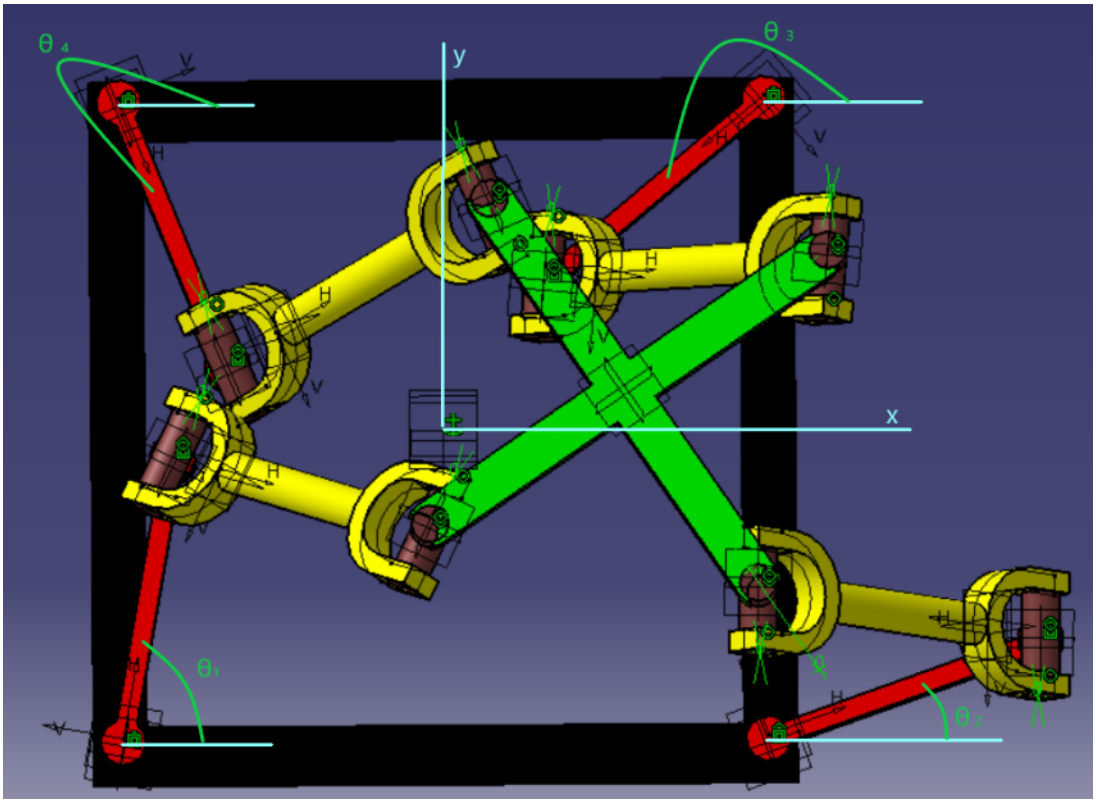


Figure 16: Adopted θ_i

7 Jacobian Matrix and Condition Number of the 4-RUU manipulator

Taking the matrices A and B of section 5, the Jacobian matrix is obtained with the following expression:

$$J = A^{-1} * B \tag{13}$$

The computation of the Jacobian once we have matrices A and B is very simple. This is the reason why in 3D manipulators is so interesting the twist and wrench analysis.
The condition number κ is a measure of the degree of independence of the columns of the manipulator's Jacobian matrix, in simpler words, how far is the mechanism from a singularity. The condition number of a Jacobian of full rank is defined as the ratio of the maximum and minimum singular values of the Jacobian.

$$\kappa = \frac{\sigma_{max}}{\sigma_{min}} \tag{14}$$

The computation of the above formulation of condition number is not simple, since singular values depend on the Eigenvalues that do not have an easy analytical expression. A computationally simpler form for calculating the condition number for a homogeneous Jacobian is given as:

$$\kappa = ||J||\,||J^{-1}|| \tag{15}$$

Equation 15 is exactly what the function **cond** of MATLAB does. It uses the matrix norm.

The condition number does not have an upper bound, $\kappa\,\epsilon\,[1, \infty]$. A condition number close to unity means a well conditioned Jacobian at that point. Manipulabity configurations for which the condition number is unity are known as Isoentropic configurations. However, it is usually useful to express the inverse of the condition number since it is bounded, $\kappa^{-1}\,\epsilon\,[0, 1]$, where a null value means that the Jacobian loses its full rank.

8 Optimization of the design variables

The optimization problem consist on obtaining the optimal values for the design variables of the manipulator. Remembering the trajectory and what these design variables are:
- R: distance between O(center of the fixed frame) and A_i, being i a value between 1 and 4(the manipulator has 4 limbs).
- r: distance between the point P and C_i.
- L1: length of leg 1.
- L2: length of leg 2.

Therefore, the design variables can be expressed as a vector:

$$x = \begin{bmatrix} R & r & L1 & L2 \end{bmatrix}$$

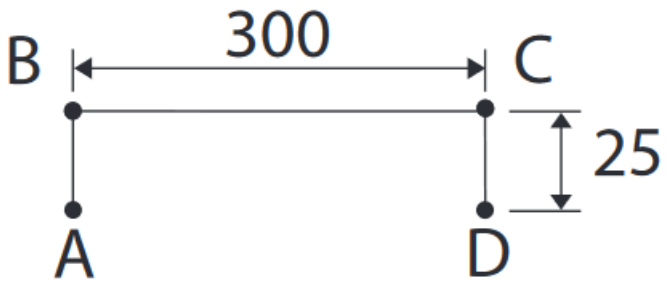


Figure 17: Path adopted for the manipulator design

The **fmincon** function of MATLAB is going to be used. This function finds the minimum of constrained nonlinear multivariable function, although it can get stack in a local minimum if the problem is not well defined, if the initial values are not realistic. This is the reason why we made use of CATIA to define the initial values which allows us to reach the points of the specified trajectory(Figure 2). These values have been found with trial and error drawing the trajectory as another part in CATIA, after parametrizing the mechanism. Find attached the EXCEL file. The values are the following:

$$R = 240mm$$

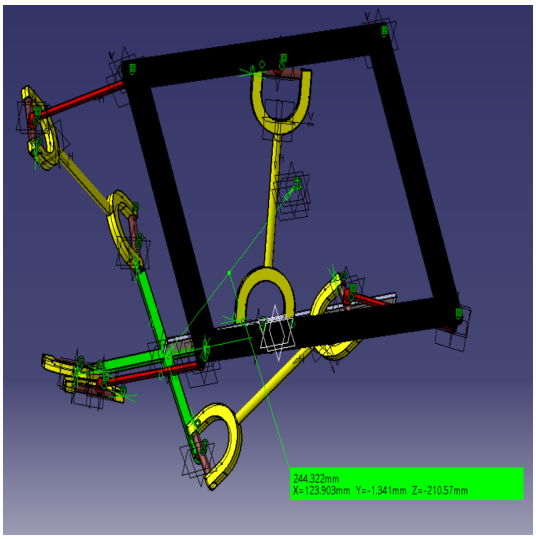
$$r = 120mm$$

$$L1 = 150mm$$

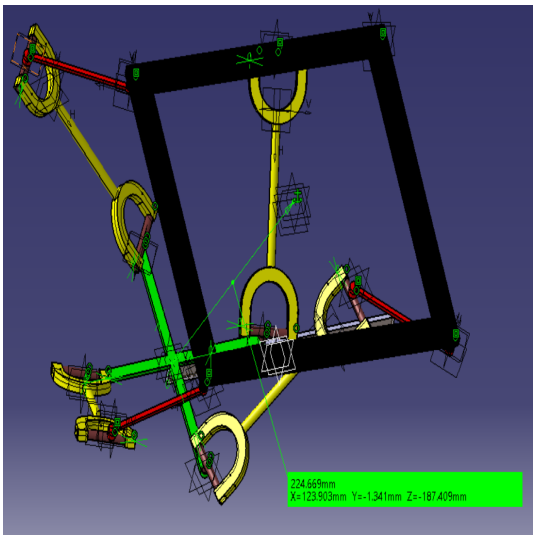
$$L2 = 260mm$$

In figure 17, it is shown how the point P is able to reach these all points. Not only the points are reachable, but also the 90º turn. In almost all points, the mechanism is very near to a serial singularity, so those values are almost the minimum ones for reaching the points of the trajectory. We could increase them just to make sure that we reach the points.

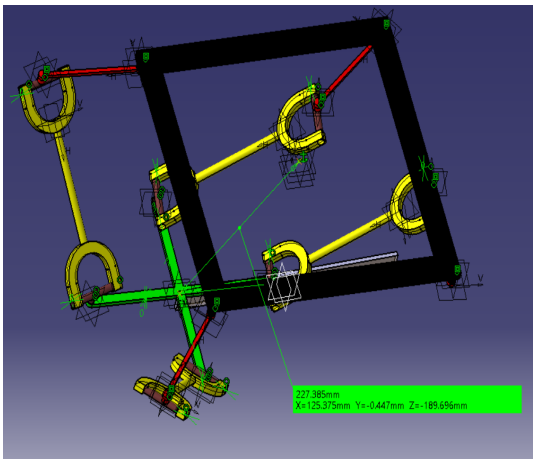
In the Table 2 is shown the actuated joints for reaching those points. It is also given the condition number. As before, we get 2 values for θ_i, and also for κ, but for this last variable, we are giving the minimum one, which probably corresponds with the real B_i point.



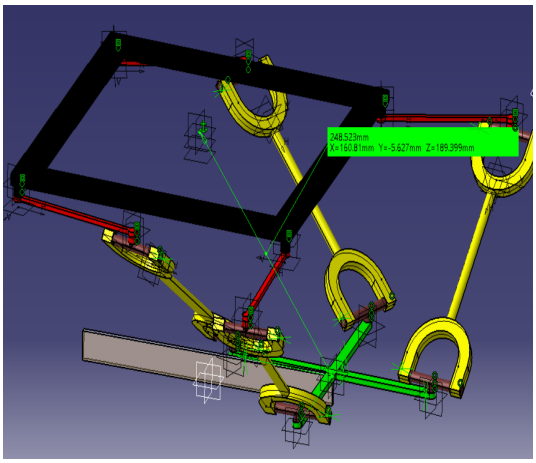
(a) Point A.



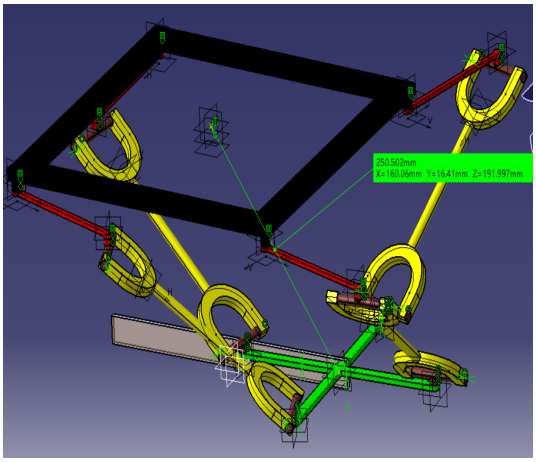
(b) Point B.



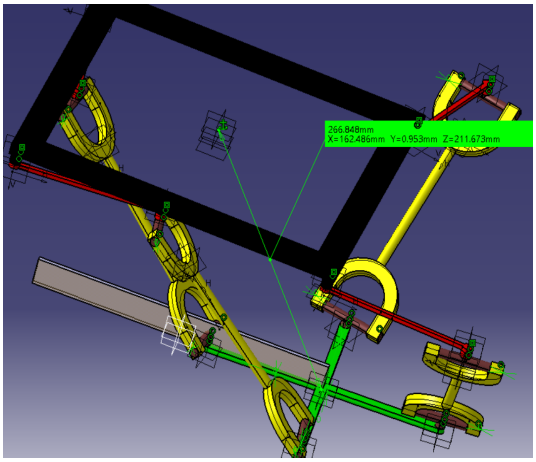
(c) Point B after turning it 90°.



(d) Point C.



(e) Point C after turning it 90°.



(f) Point D.

Figure 18: Trajectory.

$L1 = 15, L2 = 26, R = 24, r = 12[cm]$	Θ_1	Θ_2	Θ_3	Θ_4	κ
$PointA, \phi = 0^o$	$81^o/7^o$	$-31^o/3^o$	$-13^o/-60^o$	$0.5^o/-19^o$	497
$PointB, \phi = 0^o$	$87^o/-3^o$	$-59^o/27^o$	-2/-49	$18^o/-38^o$	476
$PointB, \phi = 90^o$	$37^o/-19^o$	$49^o/2^o$	$-27^o/56^o$	$3^o/-88^o$	535
$PointC, \phi = 90^o$	$89^o/-8^o$	$-81^o/-22^o$	$3^o/-60^o$	$15^o/-32^o$	415
$PointD, \phi = 0^o$	$79^o/-2^o$	$-27^o/-11^o$	$3^o/-64^o$	$4^o/-21^o$	695

Table 2: Values of θ_i for different coordinates and orientation of points of the trajectory.

The high values of κ means that the manipulator is close to a singular position. This is because we are working with almost the minimum length of the design variables for reaching those points.

The optimization problem has to be defined. As the function **fmincon** is used, some inputs have to be defined. It minimizes the objective function subject to the linear equalities $Aeq * x = beq$ and $A * x \leqslant b$. In our case we do not have these linear equalities, so A and b are defined empty. A lower and upper bounds on the design variables also has to be defined. We setted the lower bound with the previous values found in CATIA, as they were practically the minimum values to reach the points of the trajectory. The upper bounds have been setted with realistic values. Thus, the initial values x_0 for these design variables have been setted in the middle more or less of the interval.
In our case, we have non linear inequalities, as we are computing the condition number κ. These non linear inequalities $c(x)$ are defined in *nonlcon*, which is a handle function. The function **fmincon** optimizes such that $c(x) \leqslant 0$. Our inequalities constraints are defined as the inverse of the condition number(it changes in a non-linear way) at some points of the trajectory (A, B, C and D) being greater or equal to 0.1(given in the specification). Thus, the procedure for the definition of the function *nonlcon* is the following: the IGM is computed, so that we obtain the actuation values for each point of the trajectory. Then, the Jacobian matrices and finally the condition number are calculated as explained in the sections above.

Basically the optimization problem is defined as:

$$\min_{\forall x} \; f_1(x) = R + r + L1 + L$$

$$over \;\; x = [R \;\; r \;\; L1 \;\; L2]$$

$$subject\ to$$

$$\kappa_A > 0.1$$

$$\kappa_B > 0.1$$

$$\kappa_C > 0.1$$

$$\kappa_D > 0.1$$

This is exactly what it has been implemented in Matlab. However, we did not succeed in finding the right values since the script finishes saying that **fmincon** converges to an infeasible point. Thus, the final result is the lower bound, as the function tries to minimise the values.

9 Conclusion

We really enjoyed doing the project. Working with CATIA was really a great thing, and enable us to realise the utility of this program in robotics. Being able to simulate the design of a parallel manipulator was something unthinkable before the course of Mechanical Design Methods in Robotics, at Ecole Centrale Nantes, with Professor Stephane Caro.
However, we are somehow frustrated for failing in the last part of the project. We will keep working on this last part until we reach a reasonable answer.

Nothing else to say, only being thankful for your time for reading it.