Industrial Robotics Summary

Simone Zamboni

July 1, 2019

Contents

1 Introduction

We can say in general that a robot is a re-programmable machine. In our case of study a robot is fixed to a base, it is composed by bodies connected using joints and it performs action using an end effector.

1.1 Definitions

1.1.1 Types of joints

In our case we will study robots with these types of joints:

- **Prismatic joint:** a joint that only goes back and forward;
- **Revolute joint:** a joint that rotate around itself;
- **Spherical joint:** a joint that can rotate in two axis.

Moreover usually a joint has an **encoder** attached to it that send to a computer the information about the degrees or of the centimeters of displacement between the two bodies that the joint connect.
The bodies that the joints attach are called **links**, and the final part of the body is called **end effector**.

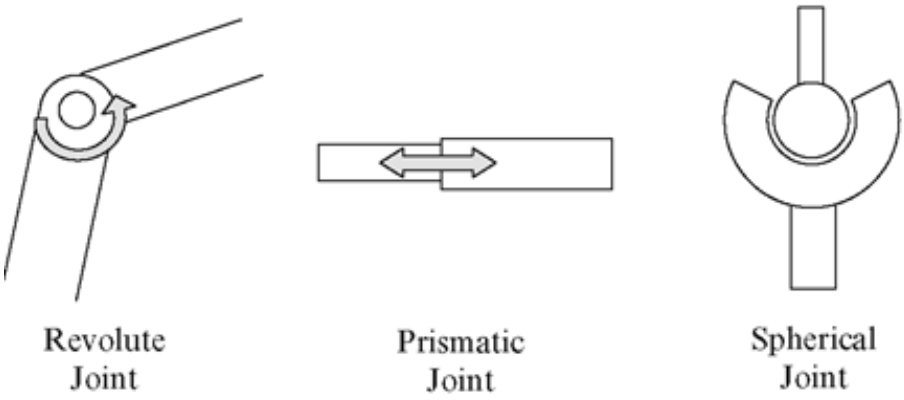

Figure 1: Types of joints illustration

1.1.2 Definition of serial, parallel and hybrid robots

Serial Robot: It's a robot composed by a set of bodies rigidly connected by joints, and these bodies form an open cinematic chain.
Parallel Robot: a robot composed by a set of rigid bodies connected by joints in a closed cinematic chain. It has more limitation than a serial robot, but it has more strength and the motors can be attached to the base
Hybrid Robot: A robot that is both serial and parallel, this type of robot is very rare.

1.1.3 Degrees of freedom

Degrees of freedom: number of independent parameters needed to describe the configuration of the system.
Every rigid body in space has 6 degrees of freedom, 3 for position and 3 for rotations, but in our case we have olonomic constraints that involve some relationships between the various coordinates of the system, so the overall number of parameters needed to describe the configuration of the system is lower.
Grubler Equation
The Gruber equation permits to calculate the number of degrees of freedom (n) for a robot with m bodies(with the base), c_1 joints with one degree of freedom and c_2 joints with two degrees of freedom:

$$n = 3 * (m - 1) - 2 * c_1 - c_2 \tag{1}$$

So for a serial robot with 4 bodies m = 5 (4 links+the base), $c_1 = 4$ for the normal revolt joints, the number of degrees of freedom is 3*(5-1) - 2*4 - 0 = 4. For a parallel robot with 4 bodies (m=5), 5 joints ($c_1 = 5$), with the Gruber equation (n) = 3.
When we have more degrees of freedom than what we need, for example a planar robot(a robot that works in a 2D environment) that needs 3 degrees of freedom (two for position and one for rotation) and we have 4 degrees of freedom, that means that for the end effector being in a certain place there are infinite possible configuration of the robot.

1.1.4 Common robot architectures

Common robot architectures are:

- Cartesian robot: a robot with three prismatic joints;
- Polar robot: a robot with two revolute joint and a prismatic joint at the end;
- Angular robot: a robot with three revolute joint.

1.2 Control of robots

In this part we discuss some of the models used to control a robot.

1.2.1 Joint Space Control Scheme

1.2.2 Work Space Control Scheme

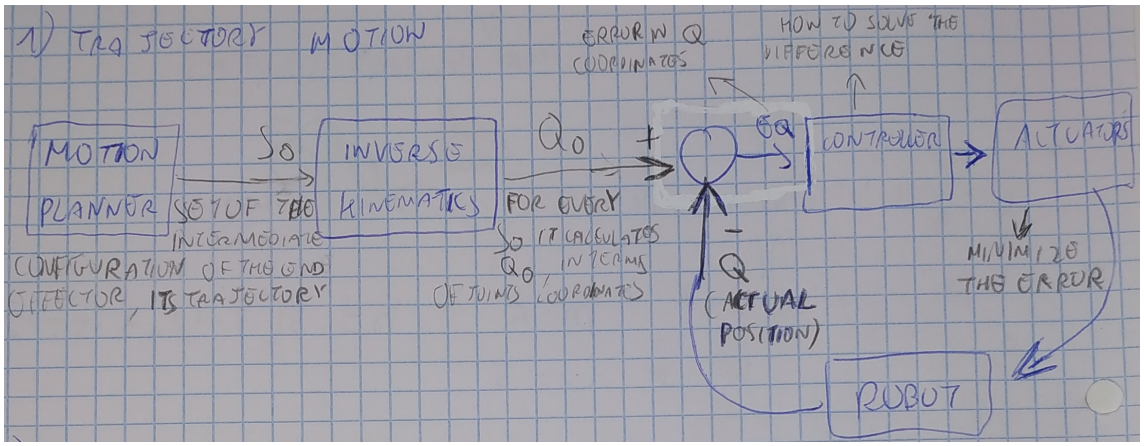

Figure 2: Joint space control model with trajectory motion

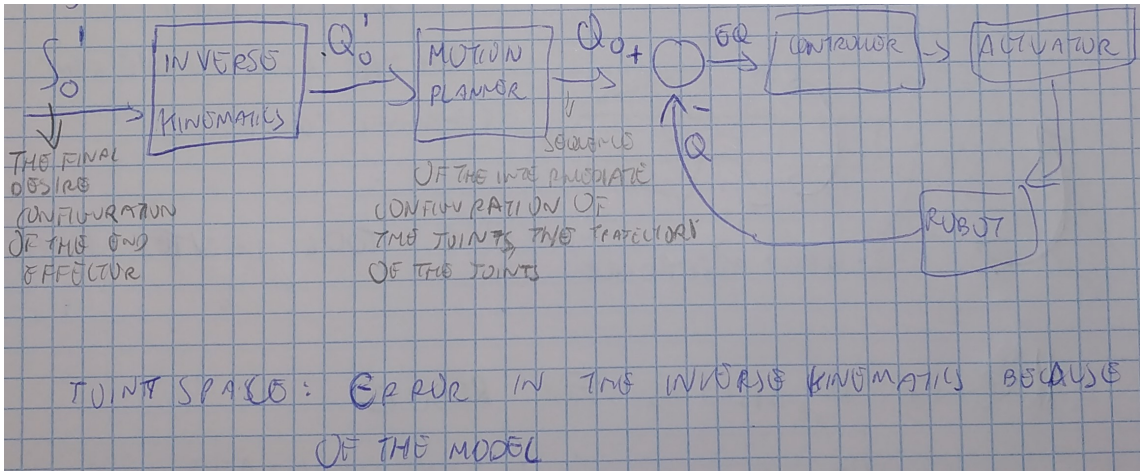

Figure 3: Joint space control model with point to point motion

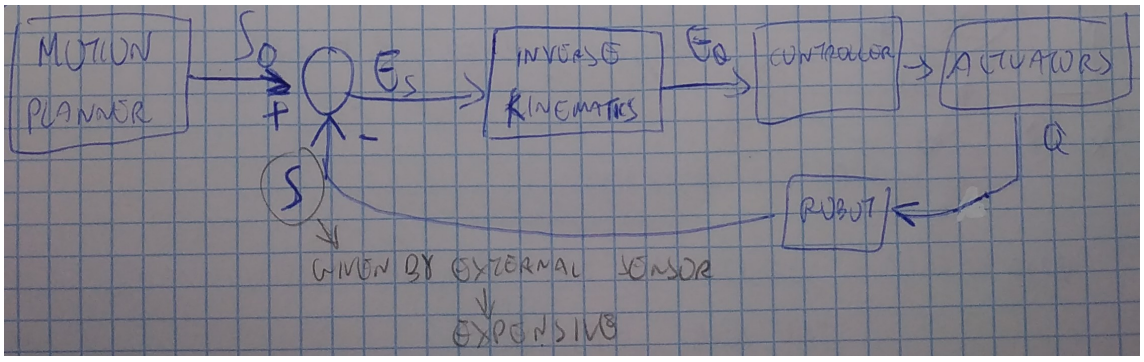

Figure 4: Ideal work space control scheme

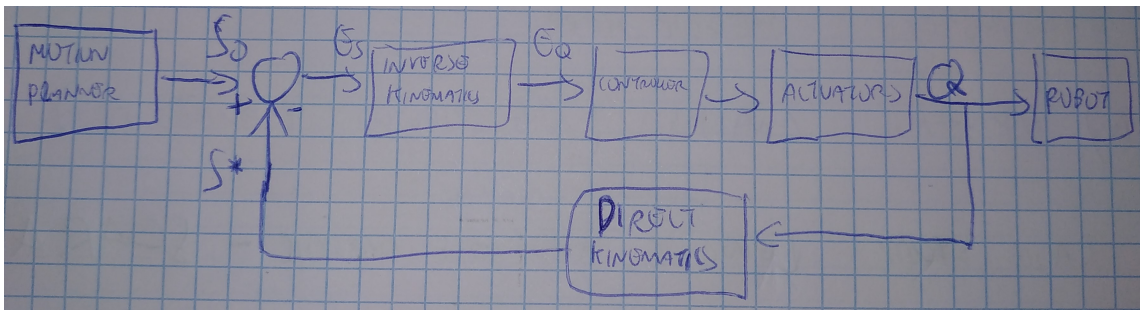

Figure 5: Actual work space control scheme

2 Static

2.1 Coordinates

Usually the degrees of freedoms of a robot coincides with the number of joints, because usually joints have motors inside. So from the top view(considering it in a 2D environment) a robot like the SCARA has two revolute joints and so two degrees of freedom, so we can write the configuration of the robot using only the two angles of the joints from the encoders inside the joints.
This is the vector of generalize coordinates in a robot (usually the values from the encoders of the joints), and it usually much smaller than all the coordinates of the system, thanks to the olonomic constraints that joints provide:

$$Q = \begin{bmatrix} q_1 \\ q_2 \\ \dots \\ q_m \end{bmatrix}$$

The vector of all the coordinates of the system (usually the position of each joint and of the end effector):

$$S = \begin{bmatrix} s_1 \\ s_2 \\ \dots \\ s_m \end{bmatrix}$$

We can write the whole configuration of the robot as a function(f) of the generalized coordinates (as always this is possible thanks to the olonomic constraints):

$$S = f(Q)$$

We can then derivate the vector of all the coordinates of the robot to have the velocity of every point of interest in the robot. These equations can be found also by derivating, using the rule of the chain, the previous formula. Remember that the derivative of a function of more variables is the Jacobian, and it has dimension nxm, with n the number of output numbers and m the number of input numbers.

$$\dot{S} = J(Q) * \dot{Q}$$

From the previous equation and the one before it we have the direkt kinematics of the robot: we know the joint coordinates and we will know the complete configuration of the robot.

2.2 Force Ellipsoid

We have now to resolve a **static problem**: a force F_s is applied to the end effector of the robot, what are the forces at the joints that balance this force?
F_q is the vector of forces applied to the joints, these are torques on a revolute joint or a linear force on a prismatic joint:

$$F_q = \begin{bmatrix} F_{q1} \\ F_{q2} \\ \dots \\ F_{qm} \end{bmatrix}$$

The **Principle of virtual work** says that for a body to stay still the virtual work must be equal to 0. Work is force times the movement of the entire system, and we can say that the virtual work is the infinitesimal work. Therefore:

$$\partial L = 0 = \partial Q^T * F_q + \partial S^T * F_s = \partial Q^T * F_q + \partial Q^T * J^T * F_s$$

$$\textit{Therefore: } F_q = -J^T * F_s$$

If we assume we have a certain force budget:

$$K_q^2 = F_q^T * F_q$$

Applying the formula $F_q = -J^T * F_s$ we obtain:

$$F_s^T * J * J^T * F_s = K_q^2$$

So F_s is proportional to K_q^2, and this relationship is contained in the A matrix defined by:

$$A = J * J^T$$

The A matrix creates an ellipsoid (when we are in the plane, in higher dimension is a similar structure) that tell us, for a specific configuration, the amount of external forces that the robot can counterbalance with only that budget. The two semi-axis of the ellipsoid represents the force direction that are not deformed.
The two semi-axis of the ellipse are given by the eigenvectors multiplied by the eigenvalues of the A matrix. The length of the two semi-axis(a,b), considering $\nu_1 and \nu_2$ the eigenvalues of the A matrix, is:

$$a = \frac{K_v}{\sqrt{\nu_1}}, b = \frac{K_v}{\sqrt{\nu_2}}$$

2.3 Velocity Ellipsoid

Let's assume we also have a velocity budget for our joints:

$$|\dot{(Q)}|^2 = \dot{Q}^T * \dot{Q} = K_v^2$$

Using the formula:

$$\dot{Q} = J^(-1) * \dot{S}$$

We obtain:

$$\dot{S^T} * (J * J^T)^(-1) * dot(S) = K_v^2 = \dot{S^T} * A^(-1) * dot(S)$$

So also in this case the velocity of the joint is proportional to the velocity budget this time by the inverse of the matrix A, and therefore we can create another ellipsoid that tell us the velocity that the robot can generate at the end effector with that velocity budget. Also the semi-axis of this ellipsoid are the eigenvectors times the eigenvalues of the inverse of the A matrix. The eigenvectors of a matrix and of its inverse are the same, so the direction of the semi-axis is the same for both the ellispoids, but the eigenvalues of a inverted matrix are 1/eigenvalues of the original matrix. Therefore the semi-axis of this ellipsoid are:

$$a = \frac{K_v}{\sqrt{\frac{1}{\nu_1}}}, b = \frac{K_v}{\sqrt{\frac{1}{\nu_2}}}$$

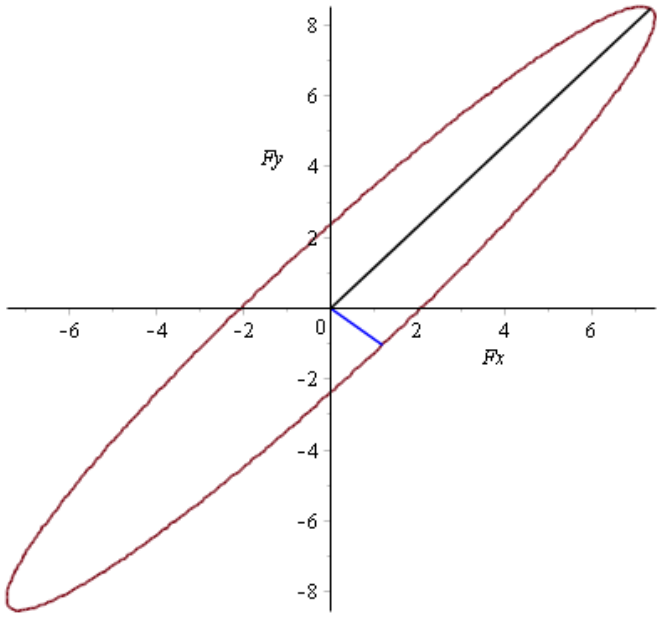

Figure 6: In a given position we calculate the A matrix, we plot the two eigenvalues multiplied by the eigenvectors, and they are the ellips semi-axis, so we can plot the whole ellipsoid. The boundaries of the ellipsoid shows the maximum force that a certain force budget can withstand

2.4 Compliance Ellipsoid

The end effector will move a little bit when we will apply a force, so we can define an appliance ellipsoid. In this case we will assume a small displacement and a linearity between the forces and the displacement. The linearity is given by a set of coefficients that we will call K_q. We also assume that this matrix has non-zero coefficients only in the diagonal ($K_q = k * I_d$) .

$$\partial Q = -K_q * F_q$$

$$\partial S = J * \partial Q =$$
$$= J * (-K_q * F_q) = J * (-K_q * (-J^T) * F_s) =$$
$$= J * K_q * J^T * F_s = J * I_d * k * J^T * F_s =$$
$$= J * J^T * (k * F_s) = A * k * F_s$$
$$k * F_s = A^{-1} \partial S$$

at this point we define a budget of compliance:

$$|(k * F_s)|^2 = K_a^2$$

Substituting in the last equation:

$$(k * F_s) * (k * F_s)^T = (A^{-1} * \partial S)^T * (A^{-1} * \partial S) =$$
$$\partial S^T * (A^{-1})^T * A^{-1} * \partial S = K_a^2$$

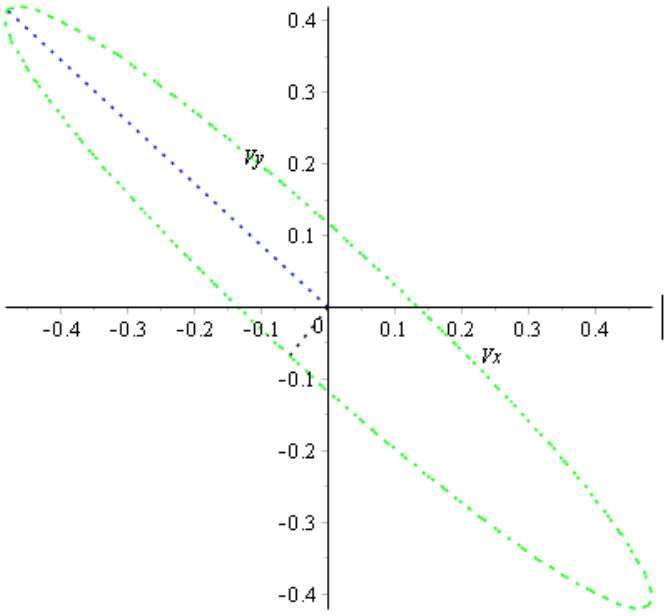

Figure 7: The robot, the position and therefore the eigenvector are the same as before, but we are considering the eigenvalues of A^-1, and therefore we have a different ellispoid with the same axis as before, but now the longest axis was the smallest in the force ellipsoid

Also in this case we have the $(A^{-1})^T * A^{-1}$ matrix that creates an ellipsoid, with dimension and axis very different from the force o velocity ellipsoid.
In the case K_q is a general matrix we define the K_s matrix say:

$$J * K_q * J^T = K_s$$

and we have that:

$$K_s * F_s = \partial S$$

we have that the compliance budget is defined simply as:

$$|F_s|^2 = K_a^2$$

Therefore:

$$K_a^2 = F_s * F_s^T = \partial S * (K_s^{-1})^T * K_s^{-1} * \partial S^T$$

So the compliance ellispoid will be given by the A_c matrix that will be: $A_c = (K_s^{-1})^T * K_s^{-1}$

2.5 Isotropy

Isotropy is defined as uniformity in all orientations, we want to measure the isotropy of a robot, with respect to the velocity but more importantly to the force ellipsoid.
To do this we calculate A, we calculate its eigenvalues and we define the condition number as follow:

$$Cond.Num = \frac{max(\nu)}{min(\nu)}$$

Ideally we want our robot to be isotrope, so with a condition number of 1. Moreover, we can calculate the isotropy for every configuration.

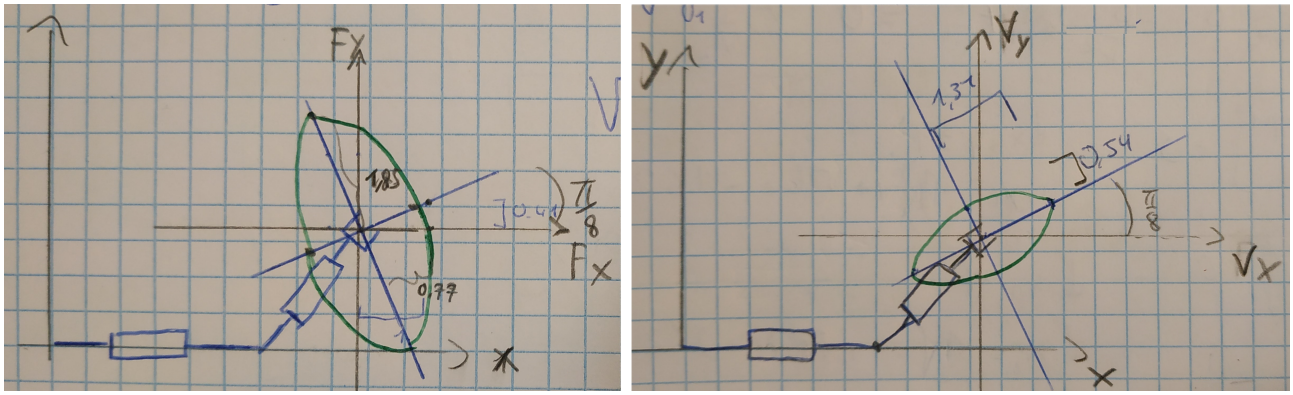

Figure 8: The force and velocity ellipsoids for the same robot. Notice that the unit of measure in the force ellipsoid are the Newtons of the force in a direction, and that the velocity ellipsoid has the same direction of the axis of the force ellipsoid, and that the unit of measure is the m/s in each direction for the velocity ellipsoid.

3 Change of reference system

3.1 Static reference systems

3.1.1 Rotation Matrix [R]

Let's take on the plane a vector $v_1 = \{v_{x1}, v_{y1}\}^T$ and rotate counterclockwise it by an angle θ, we will obtain a vector $v_2 = \{v_{x2}, v_{y2}\}^T$. The transformation from v_1 to v_2 is the following:

$$\begin{cases} v_{x2} = v_{x1} * \cos\theta - v_{y1} * \sin\theta \\ v_{y2} = v_{x1} * \sin\theta + v_{y1} * \cos\theta \end{cases}$$

We have that: $\{v2\} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} v_{x1} \\ v_{y1} \end{bmatrix}$

We can call the matrix $[R_\theta$ and is not only the matrix that when is multiplied by a vector it rotates a that vector counterclockwise by an angle θ, but this rotation can be seen as a change of reference system.

In fact if we assume v1 is a vector represented in a reference system rotated in respect to the other reference system, $[R_\theta$ is the matrix that takes v1 and gives as output v1 represented in the other reference system. If the rotated reference system is moving around the z-axis, we just change the number θ and we can change reference system to the vectors represented in the moving reference system.

We will represent a point P represented in a reference frame a as : ${}^a\{P\}$. Also the rotation matrix R that takes a point represented in a frame a and gives the same point represented in frame b will be written as ${}^b_a[R]$. Also usually frame 0 is the inertial frame fixed to the ground.

Therefore ${}^0\{P\} = {}^0_1 [R] *^1 \{P\}$

The structure of the rotation matrix is simple: it is a square matrix with dimension equal to the dimension n for the $mathbbR^n$ space. Its n columns are the vectors of the canonical base of frame a (in the previous case frame *1*) represented in frame b (in the previous case frame *0*).

If we have more rotations, for example we rotate frame *0* by θ_1 and then the new frame *1* by θ_2 obtaining frame 2, we can simply combine the various rotation matrices:

$${}^0\{P\} = {}^0_1 [R] *^1_2 [R] *^2 \{P\}$$

Therefore: ${}^0_2[R] = {}^0_1 [R] *^1_2 [R]$ Another property of the [R] matrix is that :

$$({}^0_1[R])^{-1} = ({}^0_1[R])^T = {}^1_0 [R]$$

3.1.2 Rototranslation Matrix [M]

When we have two reference systems, *0* and *1*, that differs only by a translation of a vector {t}, we can write:

$${}^0\{P\} = {}^0 \{t\} +^1 \{P\}$$

When we have a rotation and a translation together, we can write:

$${}^0\{P\} = {}^0 \{t\} + {}^0_1 [R] *^1 \{P\}$$

We can combine the information about the rotation and the translation in only one matrix, so it will all be simpler, the ${}^0_1[M]$ matrix. This matrix will be a square matrix of

dimension *n+1*, for the $mathbbR^n$ space. Its first *n+1* rows and columns are equals to the ${}^0_1[R]$ matrix, the n+1 column will be the ${}^0\{t\}$ vector, and the n+1 row will be full of zeros except for the last number that will be 1.

$$ {}^0_1[M] = \begin{bmatrix} {}^0_1[R_{1,1}] & {}^0_1[R_{1,2}] & \cdots & {}^0_1[R_{1,n}] & {}^0\{t_1\} \\ {}^0_1[R_{2,1}] & {}^0_1[R_{2,2}] & \cdots & {}^0_1[R_{2,n}] & {}^0\{t_2\} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ {}^0_1[R_{n,1}] & {}^0_1[R_{n,2}] & \cdots & {}^0_1[R_{n,n}] & {}^0\{t_n\} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} $$

In order to use this matrix, of dimension *n+1*, with points of dimension *n*, we attach the number 1 at the end of the point, in the *n+1* position of the point. In this notation we attach instead the number 0 to vectors in the *n+1* position, in order to differentiate them to the points. In this notation a Point + Vector = Point (last number 1), Vector + Vector = Vector (last number 0), Point - Point = Vector (last number 0) and Point + Point = Impossible (the last number would be 2).

With ${}^1\{P\} = \begin{bmatrix} {}^1\{P_1\} \\ {}^1\{P_2\} \\ \vdots \\ {}^1\{P_n\} \\ 1 \end{bmatrix}$ We have:

$$ {}^0\{P\} =^0_1 [M] *^1 \{P\} $$

The inverse of the M matrix is not its transpose, like the R matrix, but its inverse is not difficult. The first n rows and n columns of the M^{-1} are the $({}^0_1[R])^{-1}$, and therefore $({}^0_1[R])^T$. The last row as always is all 0 and a final 0, and the last column, in position n+1, is ${}^1\{t\}$.

3.1.3 Elicoidal-RotoTranslation Matrix [Q]

Imagine a point of a rigid body moved from position 1 to position 2. Being part of a rigid boby ${}^1\{P\} =^2 \{P\}$, but ${}^0\{P1\}! =^2 \{P2\}$, with P1 being the point in the initial position and P2 the point in the final position.
It would be nice to have a matrix that we can multiply by ${}^0\{P1\}$ and it will get us ${}^0\{P2\}$. This matrix is the Elicoidal-RotoTranslation Matrix, the Q matrix. This matrix will perform this calculation:

$$ {}^0\{P1\} = Q_0 *^0 \{P2\} $$

We know that ${}^0\{P1\} =^0_1 [M] *^1 \{P1\}$, and ${}^0\{P2\} =^0_2 [M] *^2 \{P1\}$,and for how we defined the Q matrix:

$$ {}^0\{P2\} =^0_2 [M] *^2 \{P2\} = Q_0 *^0_1 [M] *^1 \{P1\} = Q_0 *^0 \{P1\} $$

But we know that ${}^2\{P2\} =^1 \{P1\}$ for the rigid body property, and therefore:

$$ Q_0 =^0_2 [M] *^1_0 [M] $$

The Q matrix has a defined structure: in the first three rows and columns there is the rotation matrix, in the last row there are all 0 and lastly a 1, and in the last column, there

is the $\{\Delta T\}$ vector, that is the position in frame 2 of the point that in frame 1 overlaps with the origin 0.
A Q matrix has always dimension 4, because we are discussing rigid bodies in space, therefore the rotation matrix is a 3x3 matrix and the $\{\Delta T\}$ vector has 3 coordinates. If we consider {x},{y},{z} the three column vectors that form the rotation matrix of Q, that will be ${}^0_2[R] *^1_0 [R]$, we can write the general form of a Q matrix:

$$[Q]_0 = \begin{bmatrix} x_x & y_x & z_x & \{\Delta T_x\} \\ x_y & y_y & z_y & \{\Delta T_y\} \\ x_z & y_z & z_z & \{\Delta T_z\} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It is also possible to describe the movement in frame that are different than 0, for example in frame 3. To convert the Q_0 matrix from frame 0 to 3 the formula is the following:

$$Q_3 =^0_3 [M] * Q_3 *^3_0 [M]$$

3.1.4 From [Q] to u-axis and alpha angle

We can describe the action performed by the Q matrix, a movement of a rigid body from a place to another, with a rotation of an angle α around an axis formed by a unitary ($|u|^2 = 1$) vector {u} passing for a point P*, and also a translation upward of a quantity h around this axis. There is a theorem that tell us that this type of movement is unique for every possible movement in the space. These parameters can be computed using the Q matrix:

$$\sin\alpha = \pm\sqrt{(y_z - z_y)^2 + (x_z - z_x)^2 + (x_y - y_x)^2}$$

$$\cos\alpha = \frac{x_x + y_y + z_z - 1}{2}$$

Case when $\cos\alpha \neq 1$:

$$u_x = \pm\sqrt{\frac{x_x - 1}{1 - \cos\alpha} +}$$

$$u_y = \pm\sqrt{\frac{y_y - 1}{1 - \cos\alpha} +}$$

$$u_z = \pm\sqrt{\frac{z_z - 1}{1 - \cos\alpha} +}$$

$$u_x * u_z = \pm\sqrt{\frac{x_z + z_x}{2 * (1 - \cos\alpha)}}$$

$$u_x * u_y = \pm\sqrt{\frac{x_y + y_x}{2 * (1 - \cos\alpha)}}$$

$$u_y * u_z = \pm\sqrt{\frac{y_z + z_y}{2 * (1 - \cos\alpha)}}$$

$$h = \{\Delta T\} * \{u\}$$

Case when $\sin\alpha \neq 0$, therefore $\alpha \neq k * \pi$:

$$u_x = \frac{y_z - z_y}{2 * \sin\alpha}$$

$$u_y = \frac{z_x - x_z}{2 * \sin\alpha}$$
$$u_z = \frac{x_y - y_x}{2 * \sin\alpha}$$

In all the cases we have (with R the rotation matrix of the Q matrix):

$$\{P\} = \frac{1}{2 * (1 - \cos\alpha)} * (I_d - [R]^{-1}) * \{\Delta T\}$$

We can also define the pitch, the step of the translation :

$$p = \frac{h}{\alpha}$$

At the end we have a point {P} that is rotated around and axis by α and translated upward this axis by h. This axis is defined by the unitary direction {u} apply to the point {P*}. We can also say that the radius of rotation of the point {P} is the vector that starts from {P} and reach the axis of rotation in a perpendicular way in the point {P**}, and therefore the radius of rotation of the point {P} is {P-P**}.

3.1.5 From u and alpha to [Q]

For the vector {u} we can define the $[\underline{u}]$ Matrix, as following:

$$[\underline{u}] = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}$$

This matrix is special, because when we execute the multiplication $[\underline{u}] * \{v\}$, where {v} is a vector, the result will be {u} x {v}. Other properties of this matrix are: $[\underline{u}]^T = -[\underline{u}]$ and $[\underline{u}] * \{u\} = 0$.
From the angle alpha, the vector {u} with its matrix $[\underline{u}]$, the point {P*} and h we can reconstruct the Q matrix relative to that helicoidal roto-translation:

$$[R] = I_d + [\underline{u}] * \sin\alpha + [\underline{u}]^2 * (1 - \cos\alpha)$$

$$\{\Delta T\} = [I_d - R] * \{P*\} + h * \{u\}$$

If we move by an infinitesimal amount we have some different parameters:

$$\sin\alpha -> d\alpha, \alpha -> d\alpha, \cos\alpha -> 1, \{\Delta T\} -> dt, h -> dh = p * d\alpha$$

Substituting these values in the calculation of R we can see that the second part of the calculation is 0 ($(1 - \cos\alpha) = 0$, and then we expand R in the calculation for $\{\Delta T\}$, and we remain with a quantity that we call t:

$$t = (-[\underline{u}] * \{P*\} + \{u\} * p) * d\alpha$$

Expanding this we can see:

$$Q(d\alpha) = \begin{bmatrix} I_d + [\underline{u}] * d\alpha & t * d\alpha \\ 000 & 1 \end{bmatrix} = I_d + d\alpha * \begin{bmatrix} [\underline{u}] & t \\ 000 & 0 \end{bmatrix}$$

3.1.6 [L] matrix

We can call the L matrix the quantity:

$$[L] = \begin{bmatrix} [\underline{u}] & t \\ 000 & 0 \end{bmatrix}$$

And we can calculate the Q matrix using the L matrix:

$$Q(d\alpha) = I_d + d\alpha * [L]$$

We can change the reference system of the [L] matrix easily:

$${}^0[L] = {}^0_1 [M] *^1 [L] *^1_0 [M]$$

The L matrix is especially useful when we have to describe the movement of a joint, for example a revolute joint that rotates around the z-axis will be:

$${}^0[L] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

This because there is no translation, and therefore the {t} vector is all zero, and there is only the angular velocity in the z axis, so $\omega = \{0, 0, 1\}$, and therefore the matrix is very easy.
For a prismatic joint there is only translation, and if the direction of this translation is on one of the axis, for example the z-axis, we have a L matrix like this:

$${}^0[L] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

3.2 Moving reference systems

3.2.1 Velocity Matrix

We have two reference system, 0 which is fixed and inertial, and 1 which is moving in respect to 0. For a specific point in time we can convert from 1 to 0 with the [M] matrix, but we want also to calculate the velocity of a point in frame 1 respect to frame 0. For this we use the velocity matrix W that represent the velocity of frame 1 in respect to frame 0:

$${}^0_1[W] = {}^0_1 [\dot{M}] *^1_0 [M]$$

If we assume P is a point of the rigid body that is moving and has a reference frame 1 attach to it, we ca say that:

$${}^0\{\dot{P}\} = {}^0_1 [W] *^0 \{P\}$$

The velocity matrix has a very defined structure:

$${}^0_1[W] = \begin{bmatrix} {}^0_1[\dot{R}] *^1_0 [R] & {}^0\{v\} \\ 000 & 0 \end{bmatrix}$$

Where {v} is the velocity of the point of frame 1 that overlaps with the origin of frame 0. The rotation matrix of the W matrix is the $[\underline{\omega}]$ of the unitary vector $\{\omega\}$ that represents the angular velocity in all the directions:

$$[\underline{\omega}] = [\dot{R}] *_0^1 [R] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

We use the usual formula to change reference system to the velocity matrix, so we have a velocity matrix that describe a movement from a frame to another represented in the frame 1, and we want to represent this movement in frame 0:

$$^0[W] =_1^0 [M] *^1 [W] *_0^1 [M]$$

For the velocity matrix we have the **Rivals' Theorem** :

$$_2^0[W] =_1^0 [W] +_2^1 [W]$$

3.2.2 Acceleration Matrix

We can also identify an acceleration matrix that gives the acceleration of a point in frame 0 respect to frame 1:

$$^0\{\ddot{P}\} =^0 [H] *^0 \{P\}$$

It is calculated as following (the L formula is valid when q is the degree of freedom of the joint):

$$^0[H] =^0 [\dot{W}] *^0 [W]^2 =_1^0 [\ddot{M}] *_0^1 [M] =?0[L] * \ddot{q} + [L]^2 * \dot{q}^2$$

As in the velocity matrix, the last column of the acceleration matrix represent the acceleration of the point that overlaps with the origin of the frame in which the matrix is represented, considering that point as a part of the rigid body of the moving frame.
We use the usual formula to change reference system to the acceleration matrix, so we have a acceleration matrix that describe an acceleration from a frame to another represented in the frame 1, and we want to represent this acceleration in frame 0:

$$^0[H] =_1^0 [M] *^1 [H] *_0^1 [M]$$

For the acceleration matrix we can use the **Coriolis's Theorem**:

$$_2^0[H] =_1^0 [H] + 2 *_1^0 [W] *_2^1 [W] +_2^1 [H]$$

4 SCARA Static complete example

4.1 Coordinates

We have two coordinates, α and β, that are representing the encoder output of the angle of the relative joint. These are the generalize coordinates of the system.
We can also find all the coordinates of the system that we want to model:

1. x_e = the x-coordinate of the end effector;
2. y_e = the y-coordinate of the end effector;
3. x_{g1} = the x-coordinate of the center of mass of the first link;
4. y_{g1} = the y-coordinate of the center of mass of the first link;
5. x_{g2} = the x-coordinate of the center of mass of the second link;
6. y_{g2} = the y-coordinate of the center of mass of the second link;
7. α = the rotation of the first revolute joint;
8. β = the rotation of the second revolute joint.

Moreover we have other known quantities:

- L_1 = the lenght of the first link;
- L_2 = the lenght of the second link;
- g_1 = the position on the first link of the first link center of mass;
- g_2 = the position on the second link of the second link center of mass;

4.2 Direct kinematics

We can write our set of system coordinates (that we call S) as a function of our generalized coordinates (called Q):

$$S = F(Q)$$

F is a function that takes as input the generalize coordinates and gives as outputs the coordinate of the system, therefore is a function in this spaces : $\mathbb{R}^2 \to \mathbb{R}^8$
This function is the following:

$$S = \begin{cases} x_e = L_1 * \cos\alpha + L_2 * \cos(\alpha + \beta) \\ y_e = L_1 * \sin\alpha + L_2 * \sin(\alpha + \beta) \\ x_{g1} = g_1 * \cos\alpha \\ y_{g1} = g_1 * \sin\alpha \\ x_{g2} = L_1 * \cos\alpha + g_2 * \cos(\alpha + \beta) \\ y_{g2} = L_1 * \sin\alpha + g_2 * \sin(\alpha + \beta) \\ \alpha = \alpha \\ \beta = \beta \end{cases}$$

The function that we have just written is the direct kinematics, because from the generalize coordinate we compute the coordinates of the system, that represents the positions of the various points of interest in the system.

4.2.1 Jacobian

From the function of the direct kinematics we can obtain not only the positions of the points of the systems, but also the velocity, deriving the previous formula:
$\dot{S} = derivative(F(Q))$, applying the rule of the chain:

$$\dot{S} = derivative(F(Q)) * derivative(Q) = Jacobian(F(Q)) * \dot{Q}$$

The derivative of a function in multiple variable is a matrix called Jacobian, that in this case will have 8 rows (as the output variables) and 2 columns (as our input variables). The i-esimal row of the jacobian will be the expression of the i-esimal output variable first derived in respect to α and then in respect to β, and these expressions will be respectively in the first and second column.
If we multiply the jacobian by the function of the two angular velocity we will obtain the velocity of every coordinate in our system.
Remember that usually the two generalize coordinates are expressed as function of time, and therefore their velocity will be the derivative respect to the time of the equations of the two generalized coordinates.

4.3 Force ellispoid

We have now a force $F = (F_x, F_y)$ applied to the end effector. We also have a force budget $K_f^2 = F_q^T * F_q$ that we can apply to the revolute joint, and we want to know how much force our robot can withstand.
We consider now that the only coordinates in S are the coordinates of the end effector, because now we concentrate our attention on it because is the most important part of the robot:

$$Q = (\alpha, \beta), S = (x_e, y_e)$$

$$S = F(Q), F = \begin{cases} x_e = L_1 * \cos\alpha + L_2 * \cos(\alpha + \beta) \\ y_e = L_1 * \sin\alpha + L_2 * \sin(\alpha + \beta) \end{cases}$$

$$\dot{S} = Jacobian(F(Q)) * \dot{Q}$$

$$J(F(Q)) = \begin{bmatrix} -L_1 * \sin\alpha - L_2 * \sin(\alpha + \beta) & -L_2 * \sin(\alpha + \beta) \\ L_1 * \cos\alpha + L_2 * \cos(\alpha + \beta) & L_2 * \cos(\alpha + \beta) \end{bmatrix}$$

In order to do calculate the force that our robot can withstand we calculate the A matrix (that will be a 2x2 matrix, because is given by the multiplication of two 2x2 matrices) in a specific configuration, with the angles and the length of the links defined (we have set $(\alpha, \beta, L_1, L_2)$):

$$A = J(Q) * J(Q)^T$$

Now we can write the equation of the force ellipsoid, that has as variables the components of the force and that is proportional to the A matrix:

$$F^T * A * F - K_f^2 = 0$$

In order to plot the force ellipsoid we need to have the eigenvalues and the eigenvectors of the A matrix. The eigenvalues (there will be two eigenvalues, ν_1 and ν_2) are given by the formula:

$$iegv(A) = det(A - I_d) = \nu_1, \nu_2$$

The two eigenvectors (ϑ_1 and ϑ_2) are given by:

$$\vartheta_i = (A - I_d * \nu_i) * \{F\}$$

Now the two semiaxis of the ellipsoid(a and b) are:

$$a = \sqrt{\frac{1}{\nu_1}} * \vartheta_1, b = \sqrt{\frac{1}{\nu_2}} * \vartheta_2$$

Now we can plot the force ellipsoid, considering that we have to plot it on the end effector, that the axis(x and y) are parallel to the inertial reference system and that on the x-axis there is the force F_x in Newtons and on the y-axis there is the force F_y in Newtons.

4.4 Velocity ellispoid

The creation of the velocity ellipsoid is very similar to the one in the force ellipsoid. We set our coordinates S only to contain the end effector coordinates, and we have a velocity budget for the velocities at the joints: $K_v^2 = \dot{Q}^T * \dot{Q}$. We also have a vector V containing the velocity of the end effector in the inertial reference frame, ${}^0V = \{V_x, V_y\}$
We set a specific configuration, with given $(\alpha, \beta, L_1, L_2))$ values and we calculate the A^{-1} matrix, with:

$$A^{-1} = Inverse(J(Q) * J(Q)^T)$$

We can now write the equation of the velocity ellipsoid:

$$V^T * A^{-1} * V - K_v^2 = 0$$

Also here we calculate the eigenvalues and the eigenvectors, either by the normal procedure:

$$iegv(A^{-1}) = det(A^{-1} - I_d) = \lambda_1, \lambda_2$$
$$\vartheta_i = (A^{-1} - I_d * \lambda_i) * \{V\}$$

Or we can apply the properties of the matrices. These properties says that the eigenvectors of a matrix B are the same of the matrix B^{-1}, and that the eigenvalues of a matrix B are 1 over the respective eigenvalues of the matrix B^{-1}. Therefore:

$$\vartheta_i of A^{-1} = (A^{-1} - I_d * \lambda_i) * \{V\} = \vartheta_i of A$$

$$iegv(A^{-1}) = det(A^{-1} - I_d) = \lambda_1, \lambda_2 = \frac{1}{\nu_1}, \frac{1}{\nu_2}, with \nu_1 and \nu_2 = eigenv(A)$$

So we have that the semi-axis of the velocity ellipsoid(c,d) are in the same directions of the force ellipsoids, but their length is 1 over the length of that one in the force ellipsoid:

$$c = \sqrt{\frac{1}{\lambda 1}} * \vartheta_1, d = \sqrt{\frac{1}{\lambda 2}} * \vartheta_2$$
$$c = \sqrt{\nu_1} * \vartheta_1, d = \sqrt{\nu_1} * \vartheta_2$$

Now we can plot the velocity ellipsoid, considering that we have to plot it on the end effector, that the axis(x and y) are parallel to the inertial reference system and that on the x-axis there is the velocity V_x in m/s and on the y-axis there is the velocity V_y.

4.5 Compliance ellipsoid

Now we want to plot the compliance ellipsoid. Therefore the compliance matrix of the joints is given:

$$K_q = k * \begin{bmatrix} 1 & 00 & 2 \end{bmatrix}$$

We also know that there is a force F applied to the end effector that produces a movement ∂S:

$$F = \{F_x, F_y\}, \partial S = \{dx, dy\}$$

We can now define the Ks matrix:

$$K_s = J * K_q * J^T$$

Another given quantity is the compliance budget K_a:

$$K_a^2 = |F|^2 = F * F^T$$

And now we can write the equation of the ellipse:

$$\partial S * (K_s^- 1)^T * K_s^- 1) * \partial S^T = 0$$

We can see that the compliance matrix depends on the matrix $(K_s^- 1)^T * K_s^- 1)$, that we can call A_c :

$$A_c = (K_s^- 1)^T * K_s^- 1)$$

Now the procedure is the same as the A matrix, calculating the eigenvectors and the eigenvalues, and the semi-axis of the compliance ellipsoid (e,f) will be:

$$\sigma_1, \sigma_2 = eigenvalues(A_c), \varsigma_1, \varsigma_2 = eigenvectors(A_c)$$

$$e = \frac{1}{\sqrt{\sigma_1}} * \varsigma_1, f = \frac{1}{\sqrt{\sigma_2}} * \varsigma_2$$

4.6 Reference systems

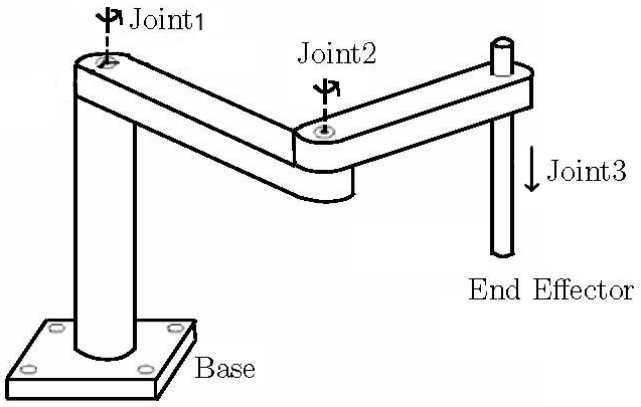

Figure 9: The SCARA robot

We now represents the various reference systems of the 3d representation of the SCARA robot:

- 0 is the inertial reference system attached to the base of the robot;
- 1 is the reference system attached to the end of link 1 in the point A (which is O_1), that moves with link 1;
- 2 is the reference system attached to the end of link 2 in the point O_2, that moves with link 2;
- 3 is the reference system attached to the end effector (the point P which is equal to O_3).

We have also some known quantities:

- h, the height from the floor of O_1 and O_2;
- L_1, the length of link 1;
- L_2, the length of link 2.

Our generalized coordinates are:

1. θ_1 : the angle of the first revolute joint;
2. θ_2 : the angle of the second revolute joint;
3. l, the length from O_2 to O_3, which is variable because is decided by the prismatic joint.

We can create the direct kinematics with the M matrices, we first write the rototranslation matrix from reference system 1 to frame 0:

$$ {}^0_1[M] = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & L_1 * \cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & L_1 * \sin\theta_1 \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

It is also easy to write the matrix from frame 2 to frame 1:

$$ {}^1_2[M] = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & L_2 * \cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & L_2 * \sin\theta_2 \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Even easier is the reference system from frame 3 to frame 2:

$$ {}^2_3[M] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -l \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

With these matrices we are able to compute the matrix from frame 3 to frame 1:

$$ {}^0_3[M] = {}^0_1[M] * {}^1_2[M] * {}^2_3[M] $$

$$ {}^{0}_{3}[M] = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & L_1 * \cos\theta_1 + L_2 * \cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & L_1 * \sin\theta_1 + L_2 * \sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & h-l \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

If we multiply this matrix by the point ${}^{3}\{0,0,0,1\}$ which is the origin of frame 3 and therefore effector, we obtain:

$$ \begin{bmatrix} L_1 * \cos\theta_1 + L_2 * \cos(\theta_1+\theta_2) \\ L_1 * \sin\theta_1 + L_2 * \sin(\theta_1+\theta_2) \\ h-l \\ 1 \end{bmatrix} $$

Which are the coordinates of the end effector in frame 0, and they represent our direct kinematics of the robot.

4.7 Velocity matrices

The SCARA robot is composed by two revolute joint and one prismatic joint, and we know the L matrices for these joints. We can say that ${}^{0}[L_{0,1}$ that is the motion of frame 1 in respect to frame 0 expressed in frame 0 is equal to ${}^{1}[L_{1,2}$, the motion of frame 2 in respect to frame 1 expressed in frame 1, that is the L matrix of a revolute joint rotating around the z-axis:

$$ [L_{rotationaroundz-axis}] =^{0} [L_{0,1} =^{1} [L_{1,2} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} $$

And we also know the ${}^{2}[L_{2,3}$ matrix, that represent the movement of frame 3 in respect to frame 2 expressed in frame 2, because is the L matrix of a prismatic joint on the z-axis:

$$ L_{translationonthez-axis} =^{2} [l_{2,3}] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} $$

From the L matrices (or the M matrices) we can calculate all the velocities matrices:

$$ {}^{0}[W_{0,1}] =^{0} [L_{0,1} * \dot{\theta}_1 =^{0}_{1} [\dot{M}] *^{1}_{0} [M] $$

$$ {}^{1}[W_{1,2}] =^{1} [L_{1,2} * \dot{\theta}_2 =^{1}_{2} [\dot{M}] *^{2}_{1} [M] $$

$$ {}^{2}[W_{2,3}] =^{2} [L_{2,3} * \dot{l} =^{2}_{3} [\dot{M}] *^{3}_{2} [M] $$

We represent every [W] matrix in the frame 0:

$$ {}^{0}[W_{1,2}] =^{0}_{1} [M] *^{1} [W_{1,2}] *^{1}_{0} [M] $$

$$ {}^{0}[W_{2,3}] =^{0}_{2} [M] *^{2} [W_{2,3}] *^{2}_{0} [M] $$

And then we can apply the Rivals' theorem:

$$ {}^{0}[W_{0,3}] =^{0} [W_{0,1}] +^{0} [W_{1,2}] +^{0} [W_{2,3}] $$

4.8 Auxiliary reference frame

The last column of the matrix ${}^0[W_{0,3}]$ is the velocity of the point of frame 3 that overlaps with the origin of frame 0. It would be easier if we have the same matrix but with instead the velocity of the origin of frame 3 (the end effector) as the last column.
We introduce therefore the auxiliary frame (called a). This reference frame has the origin on the end effector (therefore it has the same origin of frame 3), but with the axis always oriented as frame 0.
Consequently ${}^0_a[M]$ has on the rotation part the identity, and on the translation part the translation part of ${}^0_3[M]$ which is the position of the end effector respect to frame 0. We calculate the Velocity matrix of the auxiliary frame as:

$$ {}^a[W_{0,3}] = {}^a_0\,[M] * {}^0\,[W_{0,3}] * {}^0_a\,[M] $$

And this matrix will have the same rotation part as ${}^0[W_{0,3}]$, but the velocity of the end effector in the frame 0:

$$ {}^a[W_{0,3}] * \{0,0,0,1\} = {}^a\,\{O_3\} = {}^0\,\{\textit{End Effector Velocity}\} $$

The final structure of the ${}^a[W_{0,3}]$ matrix:

$$ {}^a[W_{0,3}] = \begin{bmatrix} 0 & -\dot{\theta_1} - \dot{\theta_2} & 0 & -L_1 * \sin(\theta_1) * \dot{\theta_1} - L_2 * \sin(\theta_1 + \theta_2) * (\dot{\theta_1} + \dot{\theta_2}) \\ \dot{\theta_1} + \dot{\theta_2} & 0 & 0 & +L_1 * \cos(\theta_1) * \dot{\theta_1} - L_2 * \cos(\theta_1 + \theta_2) * (\dot{\theta_1} + \dot{\theta_2}) \\ 0 & 0 & 0 & -\dot{l} \\ 0 & 0 & 0 & 0 \end{bmatrix} $$

4.9 Inverse kinematics of the position

We usually know where we want the end effector, let's say that we want the end effector in the point $E_d = (x_d, y_d, z_d)$, with a rotation ψ_d on the z axis in respect to the frame 0. We can easily construct the M matrix from this desired position to 0, that represent the rigid body of the end effector in that position:

$$ {}^0_3[M_d] = \begin{bmatrix} \cos\psi_d & -\sin\psi_d & 0 & x_d \\ \sin\psi_d & \cos\psi_d & 0 & y_d \\ 0 & 0 & 1 & z_d \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

We now want to solve the inverse kinematic problem of the position: from a desired position(S) we want to calculate the values of the generalized coordinates of the joints(Q):

$$ Q = F(S) $$

In order to do that we equate the two M matrices:

$$ {}^0_3[M] = {}^0_3\,[M_d] $$

They are two 4x4 matrices, so there will be 16 equations. In reality they are far less, because there are seven zeros and two ones in the same position of each matrix, and

therefore there are only seven equations:

$$\begin{cases} \cos\psi_d = \cos(\theta_1 + \theta_2) \\ \sin\psi_d = \sin(\theta_1 + \theta_2) \\ -\sin\psi_d = -\sin(\theta_1 + \theta_2) \\ \cos\psi_d = \cos(\theta_1 + \theta_2) \\ x_d = L_1 * \cos\theta_1 + L_2 * \cos(\theta_1 + \theta_2) \\ y_d = L_1 * \sin\theta_1 + L_2 * \sin(\theta_1 + \theta_2) \\ z_d = h - l \end{cases}$$

We can see that the first four equations are a redundant, and we can easily find from this part of the system that:

$$\theta_1 + \theta_2 = \psi_d$$

We can also easily calculate that:

$$l = h - z_d$$

With that equation we have the position of the prismatic joint for every desired configuration. We can write a new system for the equations to solve:

$$\begin{cases} \theta_1 + \theta_2 = \psi_d \\ x_d = L_1 * \cos\theta_1 + L_2 * \cos(\theta_1 + \theta_2) \\ y_d = L_1 * \sin\theta_1 + L_2 * \sin(\theta_1 + \theta_2) \end{cases}$$

We can see that we have two variables (θ_1, θ_2), with tree constraints (the three equations). Therefore this is a over-dimensioned problem. Therefore have to decide what equations satisfy, and we can choose only two equations out of three.
We can decide to satisfy the position constraint and not the rotation constraint, and we will have a system like this:

$$\begin{cases} x_d = L_1 * \cos\theta_1 + L_2 * \cos(\theta_1 + \theta_2) \\ y_d = L_1 * \sin\theta_1 + L_2 * \sin(\theta_1 + \theta_2) \end{cases}$$

Or we can decide to satisfy the x-axis position and the rotation, with this system:

$$\begin{cases} \theta_1 + \theta_2 = \psi_d \\ x_d = L_1 * \cos\theta_1 + L_2 * \cos(\theta_1 + \theta_2) \end{cases}$$

Or, finally, we can decide to satisfy the y-axis position and the rotation, with the system:

$$\begin{cases} \theta_1 + \theta_2 = \psi_d \\ y_d = L_1 * \sin\theta_1 + L_2 * \sin(\theta_1 + \theta_2) \end{cases}$$

Notice that these systems can have multiple solutions, because they are non linear equations, with sin and cos that have multiple solutions.

4.10 Inverse kinematics of the velocity

We can propose the same problem as the desired position of the end effector for the velocity of the end effector. We have a desired generalized angular velocity for the end effector ($\omega_d = (\omega_x, \omega_y, \omega_z)$ and a desired generalized linear velocity for the end effector ($v_{0,3} = (v_{0,3}x, v_{0,3}y, v_{0,3}z,)$), and we want to find the angular velocities of the revolute joints ($\dot{\theta_1}, \dot{\theta_2}$).
We can write the velocity matrix of the desired velocity (in the auxiliary reference system, so we directly have the velocity of the end effector in the last column):

$$ {}^a[W_d] = \begin{bmatrix} 0 & -\omega_z & \omega_y & v_{0,3}x \\ \omega_z & 0 & -\omega_x & v_{0,3}y \\ -\omega_y & \omega_z & 0 & v_{0,3}z \\ 0 & 0 & 0 & 0 \end{bmatrix} $$

We can now equate this matrix to the W matrix of the auxiliary reference frame:

$$ {}^a[W_d] = {}^a\,[W_{0,3}] $$

We obtain 16 equations, but in reality they are six, three from the angular velocity vector and three from the linear velocity vector:

$$ \begin{cases} \omega_x = 0 \\ \omega_y = 0 \\ \omega_z = \dot{\theta_1} + \dot{\theta_2} \\ v_{0,3}x = -L_1 * \sin(\theta_1) * \dot{\theta_1} - L_2 * \sin(\theta_1 + \theta_2) * (\dot{\theta_1} + \dot{\theta_2}) \\ v_{0,3}y = L_1 * \cos(\theta_1) * \dot{\theta_1} - L_2 * \cos(\theta_1 + \theta_2) * (\dot{\theta_1} + \dot{\theta_2}) \\ v_{0,3}z = -\dot{l} \end{cases} $$

From this system we can easily see that ω_x and ω_y are 0, in fact the robot can only rotate the end effector in the z-axis, so it will never have rotation on any other axis. Moreover also $v_{0,3}z$ is trivial, and equals to $-\dot{l}$. We therefore ramin with three equations and two variables ($\dot{\theta_1}$ and $\dot{\theta_2}$):

$$ \begin{cases} \omega_z = \dot{\theta_1} + \dot{\theta_2} \\ v_{0,3}x = -L_1 * \sin(\theta_1) * \dot{\theta_1} - L_2 * \sin(\theta_1 + \theta_2) * (\dot{\theta_1} + \dot{\theta_2}) \\ v_{0,3}y = L_1 * \cos(\theta_1) * \dot{\theta_1} - L_2 * \cos(\theta_1 + \theta_2) * (\dot{\theta_1} + \dot{\theta_2}) \end{cases} $$

This is, as before, an over-dimensioned problem, but in this case is a linear problem without multiple solutions, differently from the position inverse kinematics(??).
We have to decide what equations satisfy, and we can choose only two equations out of three. We can decide to satisfy the linear velocity constraint and ignore the angular velocity constraint, and we will have a system like this:

$$ \begin{cases} v_{0,3}x = -L_1 * \sin(\theta_1) * \dot{\theta_1} - L_2 * \sin(\theta_1 + \theta_2) * (\dot{\theta_1} + \dot{\theta_2}) \\ v_{0,3}y = L_1 * \cos(\theta_1) * \dot{\theta_1} - L_2 * \cos(\theta_1 + \theta_2) * (\dot{\theta_1} + \dot{\theta_2}) \end{cases} $$

Or we can decide to satisfy the angular velocity constraint and the linear velocity on the x-axis, and we will have a system like this:

$$ \begin{cases} \omega_z = \dot{\theta_1} + \dot{\theta_2} \\ v_{0,3}x = -L_1 * \sin(\theta_1) * \dot{\theta_1} - L_2 * \sin(\theta_1 + \theta_2) * (\dot{\theta_1} + \dot{\theta_2}) \end{cases} $$

Or, finally, we can decide to to satisfy the angular velocity constraint and the linear velocity on the y-axis, and we will have a system like this:

$$\begin{cases} \omega_z = \dot{\theta_1} + \dot{\theta_2} \\ v_{0,3}y = L_1 * \cos(\theta_1) * \dot{\theta_1} - L_2 * \cos(\theta_1 + \theta_2) * (\dot{\theta_1} + \dot{\theta_2}) \end{cases}$$

5 Denavit-Hartenberg convention

5.1 The algorithm, the parameters and the table

The Denavit-Hartenberg convention is a convention/formula/algorithm that permits to set the reference frames. We start from a reference frame *i-1*, we perform a translation on the z-axis by d_i, then on this new temporary reference system that we will call *(i-1)'*, we perform a rotation around the z'-axis by a quantity θ_i, on this new reference system that we will call *(i-1)"* we do another rotation but on the x"-axis by a quantity φ_i, on this new reference system that we will call *(i-1)"'* we do the last operation, a translation on the x"'-axis by a quantity a_i, and the resulting reference frame will be the frame *i*.
We have used four parameters in order to go from frame *(i-1)* to frame *(i)*, $d_i, \theta_i, \varphi_i and a_i$, and these parameters are called the Denavit-Hartenberg parameters.

$$^{i-1}_{i}[M] = (z_{i-1}, d_{i-1}) * (z_{i-1}, \theta_1) * (x_i, \varphi_i) * (x_i, a_i)$$

If we have on a robot a starting reference system, probably the one on the base, which is fixed, and then the reference systems of every link, we can go from one reference system to another using the Denavit-Hartenberg convention, and we can create a table with all the parameters that we have used to go from one reference system to another.

5.2 General rules

When starting we have to decide where to put the first reference frame on the robot, therefore we look forward to the first joint, and then we position out z-axis according to the type of joint. Usually the first reference frame can be a lot arbitrary, but the first joint axis and the first joint position when the encoder says 0 can be useful indicator to put the z-axis, the x-axis and the origin of the first frame, that will be fixed and will never move.
In general we look backward to put our next reference system, but if the previous reference system does not exists or it does not provide enough information we look backward.
When we have the z_{i-1} axis and the z_i axis that met in a point, this point will be the origin of frame i, and $x_i = z_{i-1} x z_i$.
When we have the z_{i-1} axis and the z_i axis that are parallel we choose the line perpendicular to the Origin of the i-1 frame and the z_i axis, and the point that this line will met with the z_i will be the origin of frame i.
Usually we try to find first z_i, then x_i and the origin of frame i, and at this point y_i is trivial, $y_i = x_i x x_i$, and therefore usually y_i is not even mentioned or draw.

5.3 Revolute joints

In the case of revolute joints the z-axis must to be positioned along the axis or rotation of the joint. Then we take the line of minimum distance between z_{i-1} and z_i, and the point where this line intersects the z_i axis will be the origin of frame i, and the continuation of this line after z_i will be the x_i axis.
For special cases (z-axis that intersecate and z-axis that are parallel), look to the general rules.

5.4 Prismatic joints

In the case of prismatic joints the z-axis is parallel to the translation axis, but can be placed anywhere. Usually we put it in a way that intersect the previous z-axis so that we can easily apply the rule of intersecting z-axis, or we put it where we want and we use the line of minimum distance rule like the revolute joint.

5.5 General M matrix

The general M matrix for going from the i frame, obtained with the DH rule, to the frame i-1 is the following:

$$ {}^{i-1}_{i}[M] = \begin{bmatrix} \cos\theta_i & \sin\theta_i * \cos\varphi_i & \sin\theta_i * \sin\varphi_i & a_i * \cos\theta_i \\ \sin\theta_i & \cos\theta_i * \cos\varphi_i & -\cos\theta_i * \sin\varphi_i & a_i * \sin\theta_i \\ 0 & \sin\varphi_i & \cos\varphi_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

5.6 Representing rotations

There are four main ways to represent rotations.
The first is to rotate on an axis, then do another rotation on the new reference frame on a different axis, and then rotate again in the new reference system. A classical example of this approach is the zyz rotation, where we rotate first on the z axis, then on the new y axis and then on the newest z axis. The problem with this approach is that id the rotation on the y axis is 0 we have a singularity, and we have infinite way to represent a rotation, and this might be a problem.
Another way, especially used in aerospace engineering is the Roll Pith Yaw approach. It is similar as before, but in this case we have three rotations, first on the z-axis, then on the original y-axis and finally on the original x-axis. The peculiarity of this rotation is that is done in respect to the original axis, but the problem is that also in this rotation there are singularities, for example a 180 degrees turn can be represented in two ways, with -Pi or with Pi.
The third method is the axis/angle representation: the rotational part of the elicoidal roto-translation matrix. We have a unitary direction {u} and an angle of rotation alpha. Also in this method we have a singularity, on 0 (infinite solutions) and on Pi (two solutions).
The only way to represent a rotation without singularity is using the Unit Quaternion method, that is a mathematical construct that starts from the axis/angle representation. If we have the direction {u} and the angle α the unit quaternion is composed by a number(η) and a vector(ε) defined as:

$$ \eta = \cos(\frac{\alpha}{2}) $$

$$ \varepsilon = \sin(\frac{\alpha}{2}) * \{u\} $$

With a specific condition:

$$ \eta^2 + \varepsilon_x^2 + \varepsilon_y^2 + \varepsilon_z^2 = 1 $$

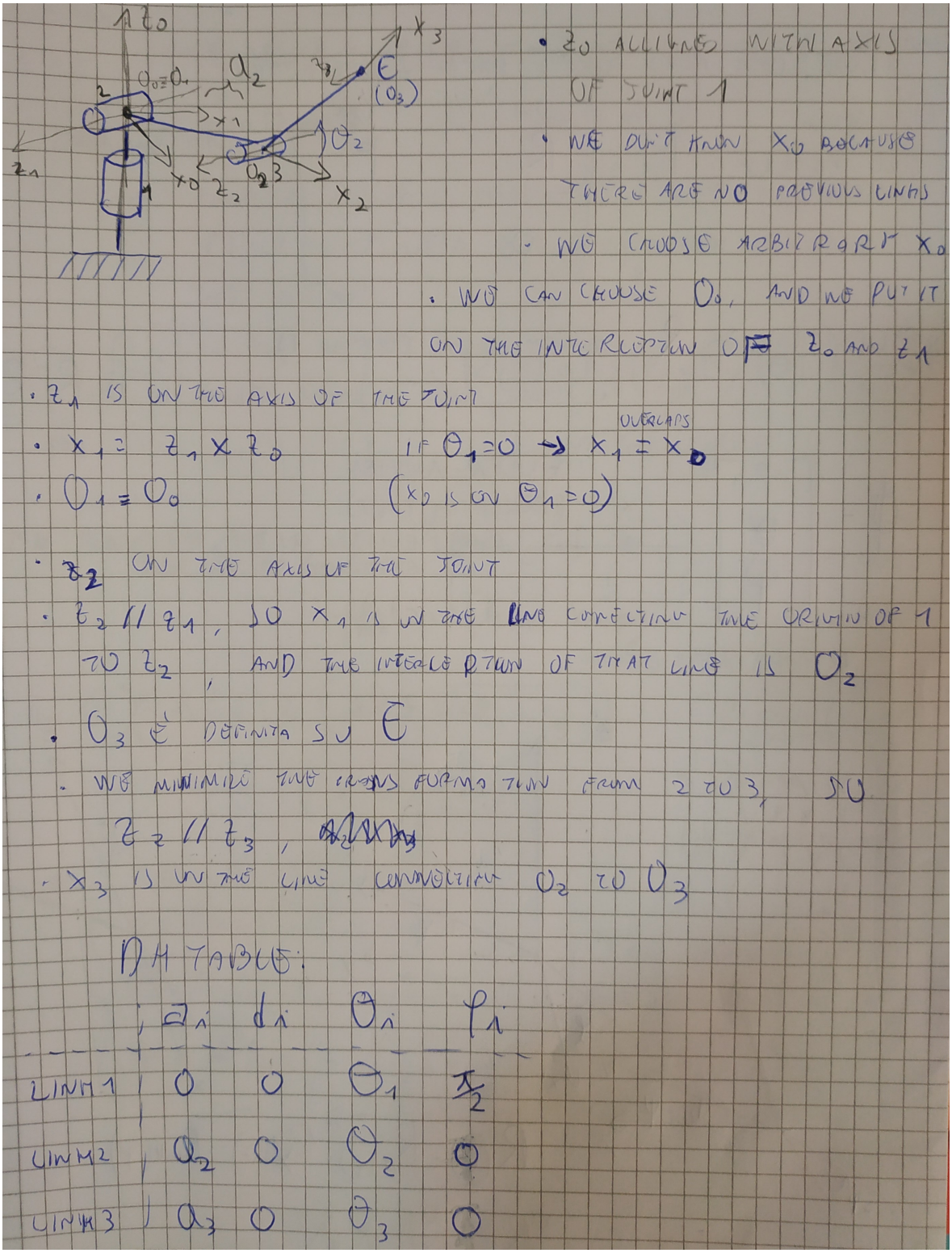

Figure 10: An application of the DH convention on a real 3D robot

6 Dynamics

6.1 Matrix of actions

We have a link on a robot and we want a mathematical instrument that permits to represents all the forces that are effecting the link. For this purpose we define the matrix of actions.
On the link there are two types of forces: linear forces applied to the origin (F) and torques(C). These forces can be represented as vectors with a component for each axis, the force on each axis and the torque present on every axis. The matrix of actions of the link i represented in the frame i is the following:

$$ {}^{i}[\Phi_i] = \begin{bmatrix} 0 & -C_z & C_y & F_x \\ C_z & 0 & -C_x & F_y \\ -C_y & C_x & 0 & F_z \\ -F_x & -F_y & -F_z & 0 \end{bmatrix} $$

Now the problem is to calculate the forces and the torques that are applied to the link. We can have a number N of force, and each force, F_j, is applied in a point ${}^{i}\{P_j\}$. Then we have the pure torque applied to the link, that we can write as the [C] matrix (considering the vector c as the sum of all the pure torques applied to the body), but in this case adding a row and a column of 0. With these assumptions this is the formula to calculate the ${}^{i}[\Phi_i]$ matrix:

$$ {}^{i}[\Phi_i] = \sum_{j=1}^{N} (\{F_j\} *^{i} \{P_j\}^T - {}^{i}\{P_j\} * \{F_j\}^T) + \begin{bmatrix} 0 & -C_z & C_y & 0 \\ C_z & 0 & -C_x & 0 \\ -C_y & C_x & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} $$

To represent the Matrix of action of frame i in a different frame, 0, we use a different way to translating it:

$$ {}^{0}[\Phi_i] =^{0}_{i} [M] *^{i} [\Phi_i] *^{0}_{i} [M]^T $$

6.2 Pseudo-Inertia Tensor [J]

This Matrix permits to us to represent the inertia of a system in an easy way. This is the structure of the matrix (for the inertial of link i represented in the frame i):

$$ {}^{i}[J_i] = \begin{bmatrix} I_{xx} & I_{yx} & I_{zx} & m * x_G \\ I_{xy} & I_{yy} & I_{zy} & m * y_G \\ I_{xz} & I_{yz} & I_{zz} & m * z_G \\ m * x_G & m * y_G & m * z_G & m \end{bmatrix} $$

Where m in the total mass of the body and x_G, y_G, z_G are the coordinates of the center of mass of the link.
Then:

$$ I_{xx} = \int x^2 \mathrm{d}m, I_{yy} = \int y^2 \mathrm{d}m, I_{zz} = \int z^2 \mathrm{d}m $$

$$ I_{xy} = I_{yx} = \int x * y \mathrm{d}m, I_{xz} = I_{zx} = \int x * z \mathrm{d}m, I_{zy} = I_{yz} = \int z * y \mathrm{d}m $$

There formulas differ from standard physics, in fact:

$$J_x = \int (y^2 + z^2)\mathrm{d}m, J_y = \int (x^2 + z^2)\mathrm{d}m, J_z = \int (y^2 + x^2)\mathrm{d}m,$$

But there is a formula that can convert from the normal quantities to the our inertial tensor:

$$I_{xx} = \frac{-J_x + J_y + J_z}{2}$$

This formula has a - on the J coordinate, and $I_{ab} = -J_{ab}$ There is also a simple formula that permits to calculate the Pseudo-Inertia tensor in the case that we have N lumped mass (m_j) that are in points P_j:

$$^i[J_i] = \sum_{j=1}^{N} (^i\{P_j\}) *^i \{P_j\}^T * m_j$$

To change the reference system in which the pseudo inertia tensor is represented we use a different formula:

$$^0[J_j] =^0_j [M] *^j [J_j] *^0_j [M]^T$$

6.2.1 Inertia of a mass lumped in a point

Let's say we have a link i, of length L, with its reference system i and its center of mass, in the middle of the link.
We now want to calculate the Pseudo-Inertia Tensor of link i. We consider that this link has all its mass concentrated in the center of mass point, that has coordinates (in frame i) : (L/2,0,0).
This link is rotating around its origin on one of the perpendicular axis to the x-axis.
Every component of the J matrix is 0 except I_{xx}, because every other coordinate is 0 except x, and therefore all other integrals in the I matrix have a multiplication by 0.
We calculate I_{xx} with the formula:

$$I_{xx} = \int_{-L}^{0} x^2 \mathrm{dm}$$

We integrate from -L to 0, because is in there that there is the link, but only in the point L/2 in the x-axis there is some mass and therefore this integral is not null. This means that this is not really an integral, but the x-coordinate of the point squared multiply by the mass of the link:

$$I_{xx} = \int_{-L}^{0} x^2 \mathrm{dm} = (\frac{-L}{2})^2 * m_i = \frac{m_i * L^2}{4}$$

6.2.2 Inertia of a mass distributed on a segment

Let's say we have a link i, of length L, and we assume that all the mass of the link is distributed along the segment representing the link on the x-axis. This segment goes from (0,0,0) to (-L,0,0), and it correspond to the physical position of the link in the reference system i.

This link is rotating around its origin on one of the perpendicular axis to the x-axis.
We now want to calculate the Pseudo-Inertia Tensor of link i, and as before the only non-null component isI_{xx}, therefore :

$$I_{xx} = \int_{-L}^{0} x^2 \mathrm{dm}$$

We can now transform an integral of the mass in an integral on the x-coordinate if we add ρ that represents the density of the mass on a line, ad if we assume ρ is constant:

$$\int_{-L}^{0} x^2 \mathrm{dm} = \int_{-L}^{0} x^2 * \rho \mathrm{dx} =_{-L}^{0} [\frac{x^3}{3}] * \rho = \frac{L^3}{3} * \rho$$

But with $\rho = \frac{dm}{dx}$ means that $\rho * L = m_i$, therefore:

$$I_{xx} = \frac{L^3}{3} * \rho = \frac{m_i * L^2}{3}$$

The J matrix in this case will be:

$${}^{i}J_i = \begin{bmatrix} \frac{m_i * L^2}{3} & 0 & 0 & m_i * \frac{-L}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ m_i * \frac{-L}{2} & 0 & 0 & m_i \end{bmatrix}$$

6.3 Lagrange approach

The Lagrange formula permits us to define the equation of the movement, using the Lagrange function.
The Lagrange function uses energy, and therefore we first have to define T, which is the kinetic energy, and U, which is the potential energy.

6.3.1 Kinetic energy of a link

In order to calculate the kinetic energy of link j we use this formula (considering that 0 is the inertial reference frame):

$$T_j = \frac{1}{2} * Tr({}^{0}[W_{0j}] *^{0} [J_j] *^{0} [W_{0j}]^T)$$

Where Tr is the trace, the sum of the elements on the diagonal of the matrix, ${}^{0}[W_{0j}]$ is the velocity matrix of frame j in respect to frame 0 represented in frame 0 and ${}^{0}[J_j]$ is the pseudo-inertia tensor of link j represented in frame 0.

6.3.2 Potential gravitational energy of a link

For the potential energy of link j usually is only the gravity, and therefore we introduce the gravity acceleration matrix (considering that 0 is the inertial reference frame):

$${}^{0}[H_{gj}] = \begin{bmatrix} 0 & 0 & 0 & a_x \\ 0 & 0 & 0 & a_y \\ 0 & 0 & 0 & a_z \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Where {a} is the acceleration of the moving point that overlaps with the origin. This matrix differs from the normal acceleration matrix because it only consider the gravity as the acceleration, an no other acceleration (so angular acceleration does not exists), therefore $|\{a\}|^2 = g$.
Usually we put an axis, for example the z-axis, parallel to the floor, so that $\{a\}$ is equal to {0,0,-g} and therefore easier to manage.
We use the gravity acceleration matrix in order to compute the potential energy of a generic link j, using this formula:

$$U_{gj} = Tr(-{}^0[H_{gj}] *^0 [J_j]$$

6.3.3 Lagrange equation

Now we can define the Lagrange function of the system:

$$\mathcal{L} = \sum_{j=1}^{N} (T_j - U_j)$$

From that we can apply the Lagrange formula in order to have the non conservative forces(f) that are effecting the system and after that the equations of the movement, considering that we have i coordinates we will have i equations:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = f_{qi}$$

6.4 Power

The power is given by the force time the velocity. In our case we have the matrix of actions and the velocity matrix, but multiplied in a specific way. In fact we define the **pseudo scalar product** between two matrices :

$$A \otimes B = A_{3,1} * B_{3,1} + A_{1,3} * B_{1,3} + A_{2,1} * B_{2,1} + A_{1,4} * B_{1,4} + A_{2,4} * B_{2,4} + A_{3,4} * B_{3,4}$$

When we apply this operator to the matrix of actions and the L matrix we have some interesting properties:

$$[\Phi] \otimes [L] = c_x * u_x + c_y * u_y + c_z * u_z + f_x * t_x + f_y * t_y + f_z * t_z$$

Moreover, if we have a revolute or prismatic joint the L matrix multiplied by the velocity of the link gives the velocity matrix, therefore:

$$[\Phi] \otimes [W] = c_x * \omega_x + c_y * \omega_y + c_z * \omega_z + f_x * v_x + f_y * v_y + f_z * v_z$$

Having forces and velocities this product gives the power generated (if both the matrix of action and the velocity matrix are in the inertial reference frame).
If the obtained power depends on $\dot{\theta}$ (our variable of the link), we have that:

$$Expression * \dot{\theta} = Power = F_\theta * velocity -> \frac{Power}{velocity = \dot{\theta}} = F_{theta}$$

And therefore we have the non conservative forces that are acting on the system, and we can put them in the Lagrange equation to have the equation of motion.

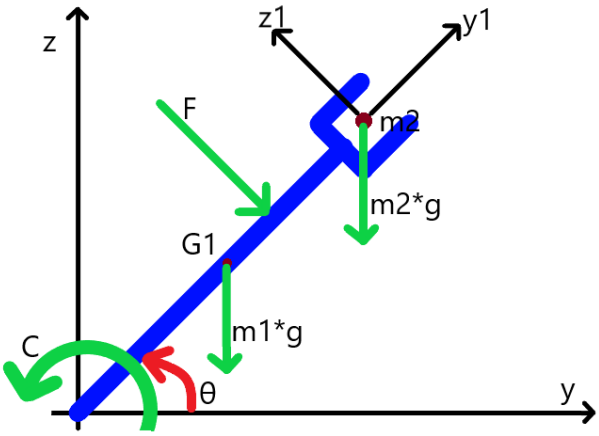

Figure 11: The example robot

6.5 Easy Example

We have a very simple robot, it is a planar robot in the y-z plane, it has only one link of lenght L, only one revolute joint in that is positioned on the origin of the inertial reference system, that has an angle θ and it applies a torque C. In this scenario we are considering dynamics and therefore forces, so we also have the position of the center of mass of link 1 (in L/2) where the gravity applies a force m1*g where m1 is the mass of the first link and g the gravity acceleration. Also at the end effector we have a payload with a mass m2, and this payload is idealized as a point, and therefore on that point is applied the force m2*g. Last we have a force F applied at $\frac{2}{3}$*L. The end effector (and therefore the payload) has t a reference system attached to it, with a rotation of θ on the x-axis respect to the origin and a translation of L.
In order to have the motion law we use the Lagrange approach. This approach requires the energy and the non conservative forces on the system.

We fist calculate ${}^1_0[M]$, that is a simple rotation around the x-axis and a translation along L:

$$ {}^0_1[M] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & L*\cos\theta \\ 0 & \sin\theta & \cos\theta & L*\sin\theta \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Then we can easily calculate ${}^0[W_{0,1}]$, with the formula with the M or the L matrix (remembering the L matrix of a revolute joint on the x-axis).

$$ {}^0[W_{0,1}] = {}^0_1[\dot{M}] *{}^1_0[M] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -\dot{\theta} & 0 \\ 0 & \dot{\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} $$

Then we can start calculating the J matrix, first for the first link.We consider link 1 as a mass distributed over a segment laying on y1, and therefore it has an inertia. To calculate the kinetic energy of link 1 fist calculate the J matrix in frame 1, that will have as non-null components only in the mass, which is m1, the I_{yy} component, that will be, as we have already calculated, $\frac{m_i*L^2}{3}$, and the y-part of the center of mass times the mass, because

the center of mass has coordinates : ${}^1(0, -\frac{L}{2}, 0)$.

$$ {}^1J_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{m_1 * L^2}{3} & 0 & m_1 * \frac{-L}{2} \\ 0 & 0 & 0 & 0 \\ 0 & m_1 * \frac{-L}{2} & 0 & m_1 \end{bmatrix} $$

We now change reference system to the J1 matrix:

$$ {}^0[J_1] = {}^0_1[M] * {}^1[J_1] * {}^0_1[M]^T $$

From this we can calculate the kinetic energy of the link:

$$ T_1 = \frac{1}{2} * Tr({}^0[W_{01}] * {}^0[J_1] * {}^0[W_{01}]^T) $$

We can repeat a similar procedure for the second mass. The reference system of the second mass is identical to the reference system of the first link, therefore ${}^0_1[M] = {}^0_2[M]$. Moreover the second mass is all lumped in a point, and therefore it has no inertia. If it would have been a line it would have some inertia and we should have calculated all the I values. One could argue that we also consider the mass of link 1 all in one point when we talk about the center of mass, but the link is still a tridimensional piece that can spin and therefore have inertia, while a mass all in one point as m2 cannot spin and therefore have inertia. Also the center of mass of m2 is in (0,0,0) in frame 2 coordinates, therefore the J matrix of m2 is very simple:

$$ {}^2J_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_2 \end{bmatrix} $$

We now change reference system to the J2 matrix (remembering that frame 1 and frame 2 are the same):

$$ {}^0[J_2] = {}^0_2[M] * {}^1[J_1] * {}^0_2[M]^T $$

It is now possible to calculate the kinetic energy of link 1:

$$ T_1 = \frac{1}{2} * Tr({}^0[W_{01}] * {}^0[J_1] * {}^0[W_{01}]^T) $$

And the kinetic energy of mass 2 (${}^0[W_{02}]$ is equal to ${}^0[W_{01}$):

$$ T_2 = \frac{1}{2} * Tr({}^0[W_{02}] * {}^0[J_2] * {}^0[W_{02}]^T) = \frac{1}{2} * m_2 * L^2 * \theta^2 $$

From the kinetic energy we move on to the potential energy of the only conservative force in the system: gravity. A given information is that the gravity falls along the z-axis, and we can therefore write the gravity acceleration matrix for both masses:

$$ {}^0[H_{g1}] = {}^0[H_{g2}] \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -g \\ 0 & 0 & 0 & 0 \end{bmatrix} $$

We have now also the information to calculate both gravitational potential energies:

$$U_{g1} = Tr(-^0[H_{g1}] *^0 [J_1 = m_1 * g * \frac{L * \sin\theta}{2}$$

$$U_{g2} = Tr(-^0[H_{g2}] *^0 [J_2] = m_2 * g * L * \sin\theta$$

Now with both kinetic energies and potential energies is possible to write the Lagrangian function (it will be a single function because we have only one generalized coordinate in this system):

$$\mathcal{L} = T_1 + T_2 - U_{g1} - U_{g2}$$

We have now to calculate the other part of the lagrangian equation, the non conservative forces on the system. In order to calculate it we need the matrix of actions of the link and the payload. No force is applied to the payload, so it not has a matrix of action. A different story is the first link, because on it there are both the torque of the revolute joint C and the external force F that are effecting it. In order to do that we can use the formula to calculate the matrix of action (considering that the vector of torques C is (c,0,0) and that we have only the force F with components(in frame 1) of (0,0,-F) and is applied in the point of frame 1 of coordinates (0,-L/3,0)):

$$^1[\Phi_1] = \sum_{j=1}^{N}(\{F_j\} *^i \{P_j\}^T -^i \{P_j\} * \{F_j\}^T) + \begin{bmatrix} 0 & -C_z & C_y & 0 \\ C_z & 0 & -C_x & 0 \\ -C_y & C_x & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$=^1 [\Phi_1] = \begin{bmatrix} 0 & -Ctot_z & Ctot_y & Ftot_x \\ Ctot_z & 0 & -Ctot_x & Ftot_y \\ -Ctot_y & Ctot_x & 0 & Ftot_z \\ -Ftot_x & -Ftot_y & -Ftot_z & 0 \end{bmatrix}$$

That gives the result:

$$^1[\Phi_1] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -C - F * \frac{L}{3} & 0 \\ 0 & C + F * \frac{L}{3} & -F & \\ 0 & 0 & F & 0 \end{bmatrix}$$

It is now possible to calculate the power of the system with the formula:

$$power =^0 [\Phi_1] \otimes^0 [W_{01}] = (-\frac{2}{3} * FL + C) * \dot{\theta}$$

From the power we can extract the force applied to the system because power = force * velocity and in our case we can simply recognize the velocity and divide the power by that:

$$\frac{power}{velocity} = force = \frac{(-\frac{2}{3} * FL + C) * \dot{\theta}}{\dot{\theta}} = -\frac{2}{3} * L * F + C = f_\theta$$

Now we have all the components and we can write the lagrangian equation:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}} - \frac{\partial \mathcal{L}}{\partial \theta} = f_\theta$$

We have found the general equation of motion of the system.
If we start putting values into the equation, with initial conditions for position and velocity, and then resolve the resulting differential equation we will found the specific equation of motion for that case, in terms of position of the arm (the θ coordinate).
We now try to put some values and see how the system behaves. First we put the motor and the force equal to 0, with initial angle at 0 and angular velocity at 0. In that case the only force on the system is the gravity, and the motion of the system will be like a pendulum, oscillating between 0 and -Pi.

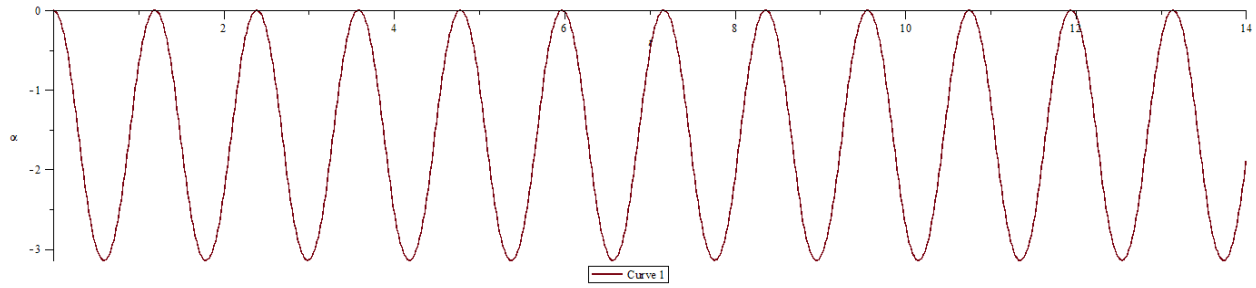

Figure 12: When the only force applied to the robot is the gravity, it behave like a pendulum, oscillating. On the x-axis there is the time and on the y-axis the angle of the revolute joint

In the case we punt a constant motor force, but not strong enough to completely counter the gravity we still have a pendulum-like movement, because the gravity is strong enough to put the arm vertical at -Pi/2, but the motor is also strong enough to lift the pendulum from there, but not enough to win the gravity when it is maximum at angle 0. Therefore the arm is in a pendulum like motion from 0 to -Pi/2.

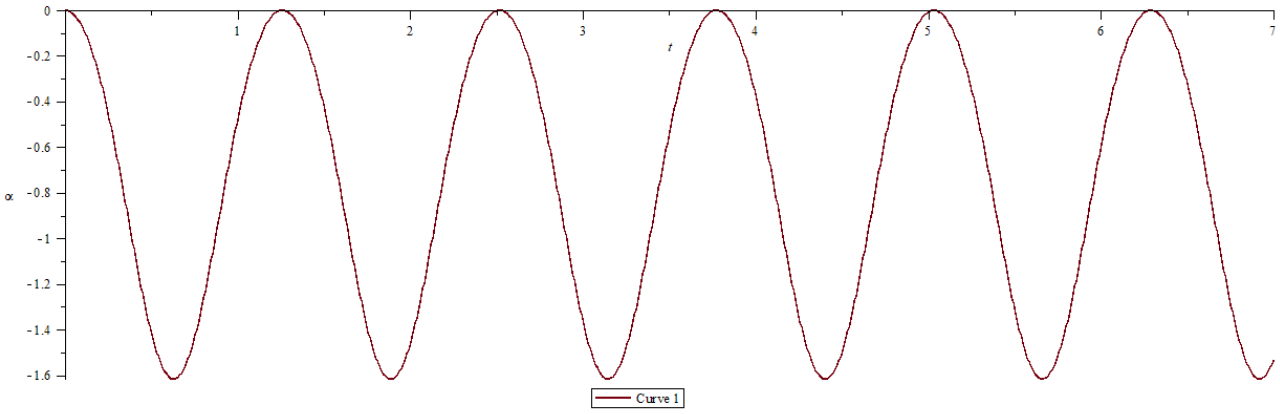

Figure 13: When the torque applied to the robot is not as powerful as the gravity, it behave like a pendulum, oscillating. On the x-axis there is the time and on the y-axis the angle of the revolute joint

If we put a stronger motor it will win the gravity force and make the arm spin increasingly faster with a constant acceleration, until the inertia of the arm will be to big to being increased by that torque.

In the last example we put a very powerful motor that act is torque in an harmonic way: $C = 70 * \sin(t)$. We also have a small force and the gravity.

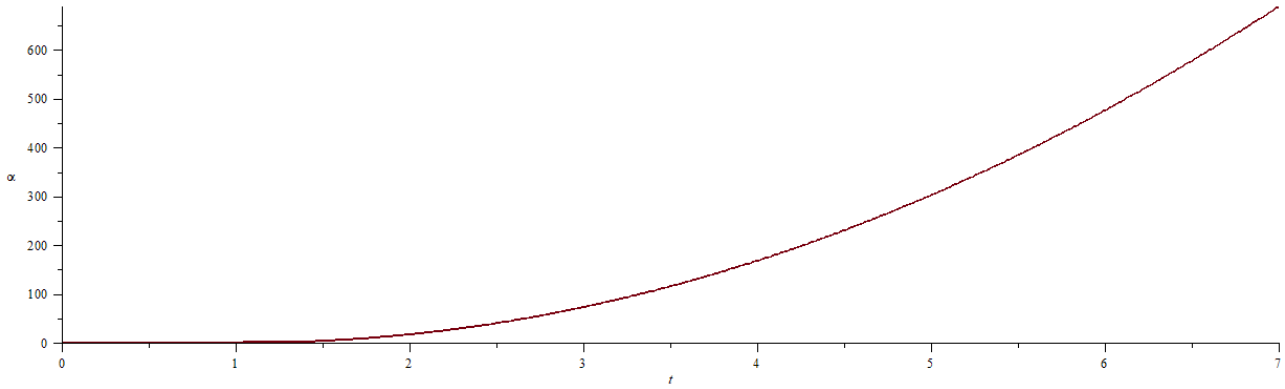

Figure 14: When the torque applied to the robot is a lot more powerful than the gravity, the arm rotates. On the x-axis there is the time and on the y-axis the angle of the revolute joint

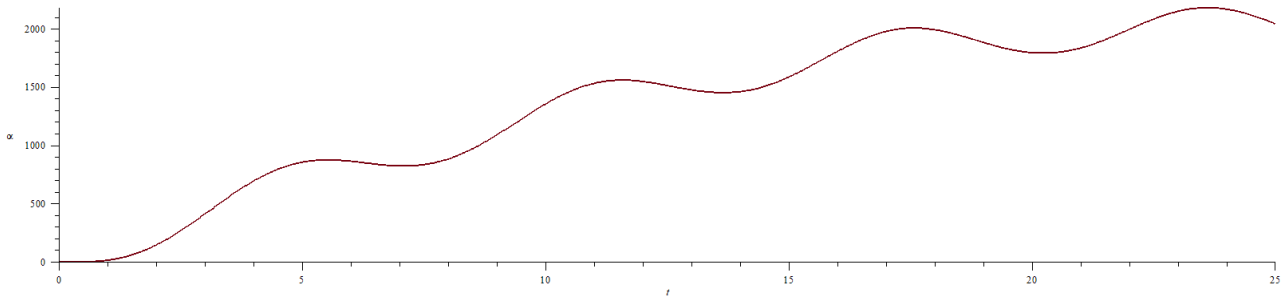

Figure 15: A very powerful motor with sinusoidal output considering gravity and a small force F. On the x-axis there is the time and on the y-axis the angle of the revolute joint

7 Maple Tricks

7.1 Kinematics

7.1.1 M matrix

M matrix of a translation of [point] and a rotation on the z axis of the angle alpha:

```
Mrotztrasl := (alpha,point) -><<cos(alpha),sin(alpha),0,0>|
<-sin(alpha),cos(alpha),0,0>|<0,0,1,0>|<point[1],point[2],point[3],1>>;
```

M matrix from frame 3 to frame 0:

```
M03 := M01.M12.M23;
```

7.1.2 Jacobian

To find the end effector position:

```
E3 := <0,0,0,1 >
E0 := M03.E3;
```

To find the Jacobian matrix (with q1(t),q2(t) and q3(t) as joint variables:

```
JS := VectorCalculus[Jacobian](subs(q1(t)=q1,q2(t)=q2,q3(t)=q3,
E0[1..3]),[q1,q2,q3]);
```

To find the determinant:

```
det := simplify(Determinant(JS));
```

Alternatively for both the Jacobian and the determinant:

```
(JS,det) := VectorCalculus[Jacobian](subs(q1(t)=q1,q2(t)=q2,q3(t)=q3,
E0[1..3]),[q1,q2,q3],'determinant'=true);
```

To find the singular configuration:

```
singularConf := solve(det=0, { q1,q2,q3 } );
```

7.1.3 Force Balance in static conditions

Let's say we are in static conditions with no movement, and we have a force (Fs := ¡Fx,Fy,Fz¿) represented in frame 0 acting on the end effector, we want to apply a force at the joints in order to balance this force:

```
Fq := - Js ^ % T . Fs ;
```

7.1.4 Velocity

To calculate the Velocity Matrix

```
W01 := simplify(map(diff,M01,t).MatrixInverse(M01));
```

H matrix :

```
H01 := simplify(map(diff,M01,t,t).MatrixInverse(M01));
```

7.1.5 Inverse Kinematics

Find the joint coordinates that produce a end effector position of (sqrt(2),0,1), with the provided data:

```
inverse_kin := solve(subs({ L1=1,L2=0.5,L3=0.3},{ E0[1]=sqrt(2),
E0[2]=0,E0[3]=1}),[q1(t),q2(t),q3(t)]);
```

The previous command will probably return more than one solution, more than one joint configuration that produces the desired end effector position. We choose the one that respect the constraints on the joint coordinate, for example one joint coordinate can only be positive or negative.

```
kinsol1 := inverse_kin[3];
```

Now we put the new configuration in a vector that we can manage, taking only the right end side of the data in kinsol1(that is the number, the left end side is the meaning, for example q1(t)):

```
kinsol:= <rhs(kinsol1[1]),rhs(kinsol1[2]),rhs(kinsol1[3])>;
```

7.1.6 L Matrix

If in link 1 we have a revolute joint on the z axis the L matrix in frame 0 will be:

```
L01 := <<0,1,0,0>|<-1,0,0,0>|<0,0,0,0>|<0,0,0,0>>;
```

If in link 2 we have a prismatic joint on the z axis the L matrix in frame 1 will be:

```
L12_1 := <<0,0,0,0>|<0,0,0,0>|<0,0,0,0>|<0,0,1,0>>;
```

Now we have to transform the L matrix from frame 1 in frame 0:

```
L12 := M01.L12_1.MatrixInverse(M01);
```

This is the matrix that we will use in order to calculate the non-conservative forces in the Lagrange equation.

7.2 Ellipses

7.2.1 Force Ellipse with 2 DOF

Create the Force Vector:

```
F:=<Fx,Fy >;
```

Create the A matrix for the force:

```
AF:=JS.(JS ^ % T);
```

Force Ellipsoid definition :

```
Fellipse := simplify(F % T.AF.F-Kf ^ 2);
```

Evaluate the A matrix for the force :

```
AF := subs( { L1=0.5,L2=0.5,Kf=1,q1=0,q2=0 } ,AF);
```

Calculate Eigenvalues and Eigenvectors :

```
(evaluesF,evectorsF):= Eigenvectors(AF);
```

Calculate the two axis of the Ellipse :

```
FEaxis1 := := simplify(1/sqrt(Re(evaluesF[1]))*[Re(evectorsF[1,1]),
Re(evectorsF[2,1])]);
FEaxis2 := := simplify(1/sqrt(Re(evaluesF[1]))*[Re(evectorsF[1,2]),
Re(evectorsF[2,2])]);
```

Plot the axis of the Force Ellipsoid:

```
with(plots):
FEaxis1plot:=plot([[0,0],FEaxis1,color=blue):
FEaxis2plot:=plot([[0,0],FEaxis2],color=green):
display(FEaxis1plot,FEaxis2plot);
```

Evaluate the Force Ellipsoid :=

```
Fellipse := subs({L1=0.5,L2=0.5,Kf=1,q1=0,q2=0},Fellipse)
```

Plot the Force Ellipsoid :

```
elliF :=plots[contourplot](Fellipse,Fx=-10..10,Fy=-10..10,contours=[0]
,grid=[100,100]):display(elliF,FEaxis1plot,FEaxis2plot);
```

7.2.2 Velocity Ellipsoid

TODO

7.3 Equation of Motion

7.3.1 Pseudo-Inertia Tensor and Kinetic Energy

Iyy of link1 with lumped mass in (x,y,z): $y^2 * m1$
Iyz of link1 with lumped mass in (x,y,z) : y*z*m1
Iyy of link1 with mass along the y-axis from 0 to -L1 : $m1 * L1^2/3$
General formula:

$$Jii = \begin{bmatrix} I_{xx} & I_{yx} & I_{zx} & m * x_G \\ I_{xy} & I_{yy} & I_{zy} & m * y_G \\ I_{xz} & I_{yz} & I_{zz} & m * z_G \\ m * x_G & m * y_G & m * z_G & m \end{bmatrix}$$

Change reference system:

```
J10 = M01.J11.(M01%T)
```

For the potential energy:

```
T1 := Trace(1/2*W01.J10.(W01 ^ % T);
```

7.3.2 Gravity Matrix and Potential Energy

My advice is to immediately write the Gravity matrix in the inertial reference frame, frame 0, where usually the gravity is in the negative z direction (in this example we write the gravity matrix of frame 1 represented in frame 0):

```
Hg10 := <<0,0,0,0>|<0,0,0,0>|<0,0,0,0>|<0,0,-g,0>>;
```

Then once you have the gravity matrix you can calculate the potential energy:

```
Ug1 := Trace(-Hg1.J10);
```

Alternatively you can find the gravity matrix of frame 1 represented in frame 1 (maybe is in the y-direction and not in the z-direction) and then change reference system to that with the usual formula:

```
Hg10 := M01.Hg11.MatrixInverse(M01);
```

7.3.3 Matrix of actions and external non conservative forces

This is the general structure of the matrix of actions:

$$Phi11 = \begin{bmatrix} 0 & -C_z & C_y & F_x \\ C_z & 0 & -C_x & F_y \\ -C_y & C_x & 0 & F_z \\ -F_x & -F_y & -F_z & 0 \end{bmatrix}$$

In order to write correctly the matrix of actions we have to take into account three factors:

1. Torque/Force used to move the current link(considered positive);
2. Reaction to the Torque/Force used to move the next link(considered negative);
3. External forces applied to the current link.

The first item, the Torque/Force used to move the current link is positive and has to be put correctly into the right spots of the matrix. The second item, the reaction to the Torque/Force used to move the next link, if it exists is negative and has to be put correctly into the right spots of the matrix. For the third item we use the formula:

$$Fext = \sum_{j=1}^{N} (\{F_j\} *^i \{P_j\}^T -^i \{P_j\} * \{F_j\}^T)$$

That in Maple for a Force=(Fx,Fy,Fz) applied in (x,y,z) is:

```
Fext := <<Fx,Fy,Fz,0>>.<<x>|<y>|<z>|<1>>
- <<x,y,z,1>>.<<Fx>|<Fy>|<Fz>|<0>>;
```

For a link with a revolute joint on the y axis and with a next link that has revolute

joint acting on the z axis of this link the first and second item on the list are in this matrix:

$$Ctot = \begin{bmatrix} 0 & C1 & C2 & 0 \\ -C1 & 0 & 0 & 0 \\ -C2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

And then for the total matrix of action we sum the two matrices.

Then we have to change reference system to the matrix:

```
Phi10 := M01.Phi11.(M02 ^ %T);
```

Then we need the L matrix of the joint in the inertial reference system. L01 is usually easy to calculate in the reference system 0, but L12 is easy to calculate in the reference system 1, so we have to:

```
L12 := M01.L12-1.MatrixInverse(M01);
```

Now we can calculate the non-conservative components in the Lagrange equation, but first we have to define the pseudo scalar product:

```
PSS := (matrix1,matrix2)->matrix1[3,2]*matrix2[3,2]+matrix1[1,3]
matrix2[1,3]+matrix1[2,1]*matrix2[2,1]+ matrix1[1,4]*matrix2[1,4]
+matrix1[2,4]*matrix2[2,4]+matrix1[3,4]*matrix2[3,4];
```

Now we can finally calculate the non-conservative components in the Lagrange equation. We do this with a pseudo scalar product between all the matrix of actions summed from the one of the current link to the one of the final link, and the L matrix of this link, for every link that we have:

```
f1 := simplify(PSS(Phi10+Phi20+Phi30,L01));
f2 := simplify(PSS(Phi20+Phi30,L12));
f3 := simplify(PSS(Phi30,L23));
```

If we have done everything correctly and there are no external actions on the link the final result should be the Torque/Force used to move the link.
If instead the L matrix we use the velocity matrix we will obtain the power of the link:

```
power1 := simplify(PSS(Phi10+Phi20+Phi30,W01));
```

7.3.4 Lagrange Equation

First we calculate the energy part of the Lagrangian equation:

```
Lagr := T1 + T2 + T3 - Ug1 - Ug2 - Ug3;
```

We will have as many equation as the number of Degrees Of Freedom that we have. To calculate the Lagrangian equations there is therefore a procedure to do for each joint coordinate:

Define a function that permits easily in Maple to perform the derivative of a function:

```
diffF := (f,x)->subs(y=x,diff(subs(x=y,f),y));
```

Then for every coordinate (in this case q1, and the non-conservative components in the Lagrange equation for this link is in f1):

```
ddq1_ Lagr := diffF(Lagr,q1(t));
ddtddq1dot_ Lagr := simplify(diff(diffF(Lagr,diff(q1(t),t)),t));
eqm1 := combine(simplify(ddtddq1dot_ Lagr-ddq1_ Lagr-f1));
```

We now have the equations of motion of the system.

7.4 Profiles

7.4.1 Base profile

In a ideal world we want to move in 1 second from the joint coordinate 0 to the joint coordinate 1, with initial and final velocity equal to 0. We do this using a function that is quadratic in time:

```
base_ profile:=piecewise(t >= 0 and t <=1/2,2*t^2,t >1/2 and t <=1
,-2*t^2+4*t-1);
```

We can easily calculate the velocity and the acceleration of this profile:

```
base_ velocity := diff(base_ profile,t);
base_ acc := diff(base_ velocity,t);
```

And plot the profile:

```
plot(base_ profile,t=0..1);
```

7.4.2 General profile

We can generalize the profile used as before, with times different than 1sec, amplitude of movement different than 1, starting position not 1 and final position not 1 and a different maximum acceleration than 4:

```
base_ profile := piecewise(t>=0 and t<=1/2*T,accmax*t^2/2, t>=1/2*T
and t<=T,accmax*T^2/4-(T-t)^ 2*accmax/2);
```

Let's say we want to go from 2 to -2 in 2 seconds with an acceration max of 2:

```
base_ profile1 := 2 -2*subs(T=2,accmax = 2,base_ profile);
```

In general if startPosition <finishPosition :

```
base_ profileN := startPosition + subs(T=Tmovement,accmax=maxAccLinkN);
```

Else if startPosition >finishPosition:

```
base_ profileN := startPosition - subs(T=Tmovement,accmax=maxAccLinkN);
```

We can then plot the profile of the joint movement:

```
plot(base_ pfofile1,t=0..Tmovement);
```

7.4.3 Time for a movement

We want to go from joint coordinate [a1,b1,c1] to joint coordinate [a2,b2,c2]. The difference between the two positions is [delta1,delta2,delta3]. We want, given the maximum acceleration of every joint [amax1,amax2,amax3], the time needed to perform the movement. In order to do that we create the function MovementTime :=

```
MovementTime := (delta,amax) ->sqrt(4*delta/amax);
```

We find the movement time of every joint :=

```
MovTime1 := evalf(MovementTime(delta1,amax1);
MovTime2 := evalf(MovementTime(delta2,amax2);
MovTime3 := evalf(MovementTime(delta3,amax3);
```

We want all three joints to finish the movement together, therefore we take the longest time and so that joint will perform the movement with its maximum acceleration, but the other joints will perform the movement in the same longest time but not with their maximum acceleration, but with a rescaled acceleration (the biggest movement time is 1 in this example):

```
Tmov := MovTime1;
```

Acceleration rescaling function:

```
RescaledAcc := (delta,time) ->4*delta/time^2;
```

Find the rescaled accelerations:

```
accR1 := RescaledAcc(delta1,Tmov);
accR2 := RescaledAcc(delta2,Tmov);
accR3 := RescaledAcc(delta3,Tmov);
```

We can now define the profiles of the movement of each joint (assuming a1,b1,c1 smaller than a2,b2,c2):

```
base_ profile1 := a1 + subs(T=Tmov,accmax=accR1,base_ profile);
base_ profile2 := b1 + subs(T=Tmov,accmax=accR2,base_ profile);
base_ profile3 := c1 + subs(T=Tmov,accmax=accR3,base_ profile);
```

7.5 Movement Control

7.5.1 Motor force for a profile

We have a desidered joint coordinate profile (profile1), the equation of motion for that joint coordinate (eqm1) and the data of the problem (link length and masses memorized in the data variable). We now want to find the motor torque/force(C1 in this case) that has to be commanded in order to follow that profile:

```
eq1M := solve(subs(data,eqm1),C1);
eq1Mplot := evalf(subs(q1(t)=profile1,q2(t)=profile2,
q3(t)=profile3,eq1M));
plot(eq1Mplot,t=0..Tmov);
```

7.5.2 Proportional Controller

We have the equations of motion of the system (eqm1,eqm2), the problem data (masses and link length in the data variable), the profile of the joints movement (profile1 and profile2), and we want to use a proportional controller to command the motors (C1 and F2) in order to perform the movement. A proportional controller is simply a constant(K1 and K2) that multiply the difference between the desired position (profile1 or profile2) and the actual joint position (q1(t) or q2(t)):

```
eq1controlled:=subs(data,K1=5000,subs(C1=K1*(profile1-q1(t)),eqm1));
eq2controlled:=subs(data,K2=5000,subs(F2=K2*(profile2-q2(t)),eqm2));
```

Now we have the equations of the controller into the equation of motion, and we have to solve the differential equation of the system, using the initial configuration, that is the start position and velocity of each joint:

```
qcontrolled:=dsolve({eq1controlled,eq2controlled,q1(0)=q1start
,q2(0)=q2start,D(q1)(0)=0,D(q2)(0)=0},{q1(t),q2(t)},
numeric,output=listprocedure);
```

We now extract the correct equations from the solution:

```
q1controlled:=qcontrolled[2];
q1controlled:=rhs(q1controlled);
q2controlled:=qcontrolled[4];
q2controlled:=rhs(q2controlled);
```

And we plot them in respect to the ideal profiles:

```
plot([q1controlled(t),profile1],t=0..Tmov);
plot([q2controlled(t),profile2],t=0..Tmov);
```

8 Typical Questions

These are some of the typical questions of an exam:

1. Find the position of the end effector in frame 0 (find M03);
2. Find the velocity and the acceleration matrix of frame 3 in respect to the fixed frame;
3. Find the Jacobian Matrix and the singular configuration (det(Js)= 0), or the Jacobian matrix of every link center of mass;
4. Find the forces at the joins needed in order to balance a force Fs in static conditions (Fq=-Js% T . Fs), or the forces needed in order to balance the self-weight of the links;
5. Find and plot the velocity and force ellipsoids, or the joint velocities needed to produce a velocity on a certain direction;
6. Find the joint coordinates that produces a position of the end effector [a,b,c] (E[1]=a,E[2]=b,E[3]=c) (also the joint velocities producing and end effector velocity?);
7. Find the motion profile of the joints from [a1,b1,c1] to [a2,b2,c2] with synchronized movements (this means find all the times, the longest time and rescale the accelerations of the joints);
8. Write the equations of motion of the system, considering a force acting on link N, represented in frame N as (fx,fy,fz);
9. Plot the torque values in order to produce the desired joint profiles, clarify the scale on the axes);
10. Find the equation of a proportional controller that produces the desired joint profiles.