Dynamics of Multi-Body System's Final Project - Problem 12

Michele Tartari

January 17, 2019

Abstract

This paper represents the report for the Dynamics of Multi-Body System's project developed on Winter Semester 2018-2019 and during which the kinematic analysis has been performed upon a given mechanism.

Contents

1. Intro

This project concerns the analysis of a mechanism composed of 8 rotational joint and 2 prismatic joint. The system has driving constraints on the prismatic joint. The driving equation are of the form:

$$x_k = l_k + a_k \sin(\omega_k t + \phi_k)$$

For this mechanism they have the following values:

Table 1: Parameters of the driving constraints

Joints	l_k	a_k	ω_k	ϕ_k
5-6	$\|M_0 - N_0\|$	0.25	2	0
7-8	$\|H_0 - G_0\|$	−0.30	5	0

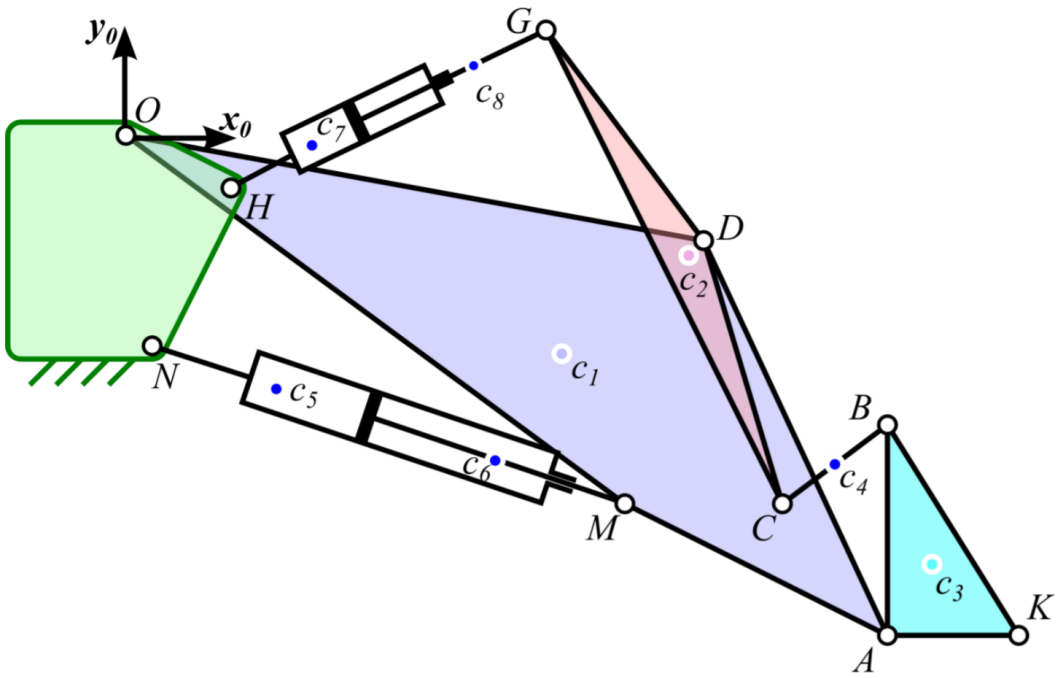

Figure 1: Drawing of the mechanism.

Table 2: Global coordinates of characteristic points (initial configuration)

	O	H	N	M	G	D	C	B	A	K
$x_{[m]}$	0.0	0.4	0.1	1.9	1.6	2.2	2.5	2.9	2.9	3.4
$y_{[m]}$	0.0	–0.2	–0.8	–1.4	0.4	–0.4	–1.4	–1.1	–1.9	–1.9

2. ADAMS Model

For this multibody system it has been realized an ADAMS model that will be used to verify the MATLAB model described in section 3. In fig.2 is visible the model of the mechanism. From this model position, velocity and acceleration where extracted to be later compared with the one obtained through MATLAB.

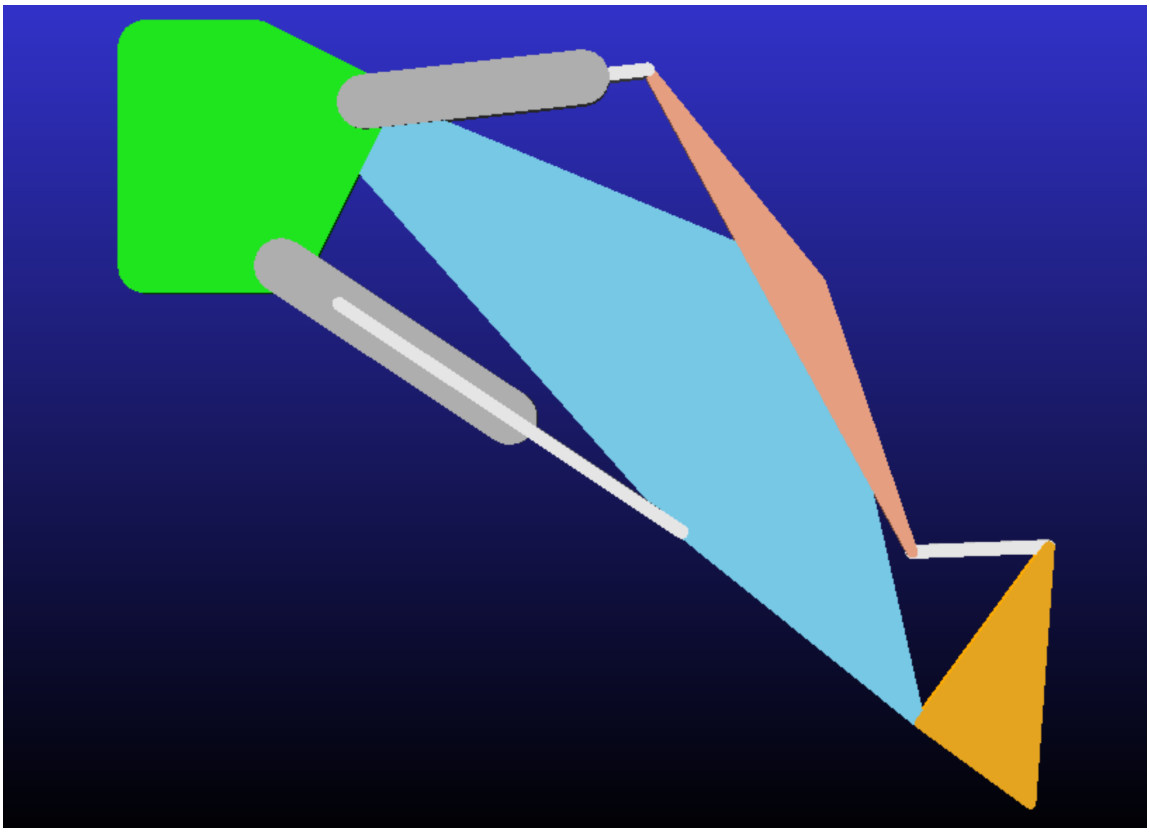

Figure 2: ADAMS model of the mechanism.

3. MATLAB Model

3.1. Notation

For clarity the following notation and definitions will be used:

- F_0 is the global frame.
- $F_i = \begin{bmatrix} x_i \\ y_i \\ \varphi_i \end{bmatrix}$ with $i \in N$ is the local frame attached to the body i.
- $R_i = \begin{bmatrix} \cos(\varphi_i) & -\sin(\varphi_i) \\ \sin(\varphi_i) & \cos(\varphi_i) \end{bmatrix}$ is the rotation matrix related to F_i.
- $r_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$ is the origin of F_i described in global frame.
- $s_P^{(i)} = \begin{bmatrix} s_{P,x}^{(i)} \\ s_{P,y}^{(i)} \end{bmatrix}$ described the position of a point (in this example P) in the local frame F_i.

- $P = r_i + R_i \cdot s_P^{(i)}$ point (in this example P) described in the global frame.
- $u^{(j)} = \dfrac{P_1 - P_2}{\|P_1 - P_2\|}$ is the unitary vector along axis of translation of a prismatic joint between the bodies i and j, and having axis translation passing through the points P_1 and P_2. $u^{(j)}$ is described in the local frame F_j. In the code we'll find it as:

```
u56 = (N_0-M_0)/norm(N_0-M_0);
u78 = (H_0-G_0)/norm(H_0-G_0);
```

- $v^{(j)} = -\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot u^{(j)}$ is the unitary vector perpendicular to the axis of translation of a prismatic joint between the bodies i and j, and having axis translation passing through the points P_1 and P_2. $v^{(j)}$ is described in the local frame F_j. In the code we'll find it as:

```
v56 = [0 1;-1 0]*u56;
v78 = [0 1;-1 0]*u78;
```

- $q = \begin{bmatrix} F_1^T, F_2^T, F_3^T, F_4^T, F_5^T, F_6^T, F_7^T, F_8^T \end{bmatrix}^T$ is the configuration vector.

3.2. Local Frame Placement

To obtain a valid multibody model it is important to define a local frame for each body. Fig.3 and table 3 shows where they have been placed, it must be noted that only the only the direction of x axis is described since the y axis will be rotated equal tot the x axis rotated anticlockwise of 90°.

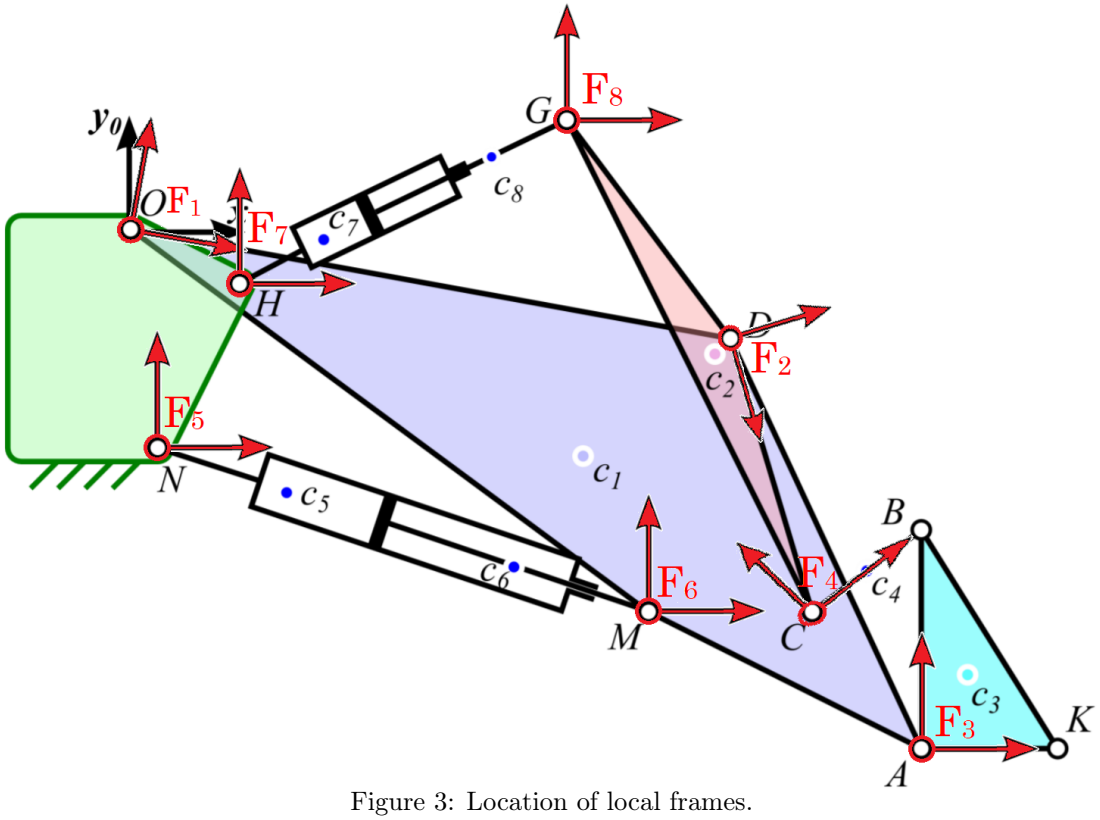

Figure 3: Location of local frames.

	Origin	Direction of x axis
F_1	O	Pointing toward D
F_2	D	Pointing toward C
F_3	A	Pointing toward K
F_4	C	Pointing toward D
F_5	N	Parallel to x_0
F_6	M	Parallel to x_0
F_7	H	Parallel to x_0
F_8	G	Parallel to x_0

Table 3: Location of local frames

3.3. Constraints' Vector

The action of a joint over the system can be described by the introduction of contraint equations. For a revolute joint connecting the bodies i, j and with axis in the point P, those equations can be written in vector form as:

$$r_i + R_i s_P^{(i)} - \left(r_j + R_j s_P^{(j)}\right) = 0_{2\times 1}$$

The constraint equations for prismatic joints can be written as:

$$\begin{bmatrix} \varphi_j - \varphi_i - \varphi_0 \\ -(R_j v)^T \cdot (r_j - r_i - R_i s_A^{(i)}) + {v^{(j)}}^T s_B^{(j)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Similarly the influence of driving equation can be described by driving constraints. For each driving equation we will have:

$$\left(R_j u^{(j)}\right)^T \cdot \left(r_j - r_i - R_i s_A^{(i)} + R_j s_B^{(j)}\right) - f(t) = 0$$

For convenience we can group the left-side part of the previously described equations in a Φ vector, also known as constraint vector.
For the K^{th} prismatic joints, between bodies i and j and axis passing trough point P, the element of the Φ vector can be written as:

$$\Phi^{K\cdot} = r_i + R_i s_P^{(i)} - \left(r_j + R_j s_P^{(j)}\right)$$

For the rotational joints that connect a body j we will have $R_i s_P^{(i)} \dot{\varphi_i}^2 = 0$. In the Φ vector it will appear at the K^{th} line (starting counting from the last line of prismatic joints).

$$\Phi = \begin{bmatrix} \vdots \\ \Phi_{2\times 1}^{K\cdot} \\ \vdots \end{bmatrix} \quad \begin{matrix} \vdots \\ K_{2\times 1} \\ \vdots \end{matrix}$$

If we look at the code we will find:

```
% Rotational Joints
Constr( 1 :  1+1) = F0(1:2)+Rot0*S0o -( F1(1:2)+Rot1*S1o );
Constr( 3 :  3+1) = F0(1:2)+Rot0*S0h -( F7(1:2)+Rot7*S7h );
Constr( 5 :  5+1) = F0(1:2)+Rot0*S0n -( F5(1:2)+Rot5*S5n );
Constr( 7 :  7+1) = F1(1:2)+Rot1*S1d -( F2(1:2)+Rot2*S2d );
```

```
Constr( 9 :  9+1) = F1(1:2)+Rot1*S1a -( F3(1:2)+Rot3*S3a );
Constr(11 : 11+1) = F1(1:2)+Rot1*S1m -( F6(1:2)+Rot6*S6m );
Constr(13 : 13+1) = F2(1:2)+Rot2*S2g -( F8(1:2)+Rot8*S8g );
Constr(15 : 15+1) = F2(1:2)+Rot2*S2c -( F4(1:2)+Rot4*S4c );
Constr(17 : 17+1) = F3(1:2)+Rot3*S3b -( F4(1:2)+Rot4*S4b );
```

For the K^{th} prismatic joints, between bodies i and j, the element of the Φ vector can be written as:

$$\begin{aligned}\Phi^{K\angle} &= \varphi_j - \varphi_i - \varphi_0 \\ \Phi^{K\uparrow} &= (R_j v^{(j)})^T \cdot (r_j - r_i - R_i s_A^{(i)}) + v^{(j)\,T} s_B^{(j)}\end{aligned}$$

Thus, in the Γ vector they will appear at the K^{th} couple of lines (starting counting from the last line of rotational joint).

$$\Phi = \begin{bmatrix} \vdots \\ \begin{bmatrix} \Phi^{K\angle} \\ \Phi^{K\uparrow} \end{bmatrix}_{2\times 1} \\ \vdots \end{bmatrix} \quad \begin{matrix} \vdots \\ K_{2\times 1} \\ \vdots \end{matrix}$$

If we look at the code we will find:

```
% Translation Joints
theta0_56= F6_0(3)-F5_0(3);
theta0_78= F8_0(3)-F7_0(3);
Constr(19 : 19+1) = [F6(3)-F5(3)-theta0_56; (Rot5*v56)' *(F5(1:2) -F6(1:2) -Rot6*S5n) +v56'*S6m];
Constr(21 : 21+1) = [F8(3)-F7(3)-theta0_78; (Rot7*v78)' *(F7(1:2) -F8(1:2) -Rot8*S7h) +v78'*S8g];
```

Finally the remaining lines of the Φ vector will be filled by the driving constraints. For the D^{th} driving constraint, between bodies i and j, the element of the jacobian matrix can be written as:

$$\Phi^{D\uparrow} = \left(R_j u\right)^T \cdot \left(r_j - r_i - R_i s_A^{(i)} + R_j s_B^{(j)}\right) - f(t)$$

Thus, in the Φ vector it will appear at the D^{th} line (starting counting from the last line of prismatic joints).

$$\Phi = \begin{bmatrix} \vdots \\ \Phi^{D\uparrow}_{1\times 1} \\ \vdots \end{bmatrix} \quad \begin{matrix} \vdots \\ D_{1\times 1} \\ \vdots \end{matrix}$$

If we look at the code we will find:

```
% Driving Equations
Constr(23)= (Rot5*u56)'*(F5(1:2) -F6(1:2) +Rot5*S5n -Rot6*S6m) -(norm(M_0 -N_0) -0.25*sin(2*t));
Constr(24)= (Rot7*u78)'*(F7(1:2) -F8(1:2) +Rot7*S7h -Rot8*S8g) -(norm(G_0 -H_0) +0.30*sin(5*t));
```

3.4. Jacobian Matrix

The jacobian matrix defines the variation of quantities with relation to the variation of the $[q, t]$ variables. It is useful to divide the effect of q from those of t, we will thus define respectively a Φ_q

matrix and a Φ_t vector. For the K^{th} Rotational joints, between bodies i and j, the element of the jacobian matrix can be written as:

$$\begin{aligned}
\Phi^{K\cdot}_{r_i} &= I_{2\times 2} \\
\Phi^{K\cdot}_{r_j} &= -I_{2\times 2} \\
\Phi^{K\cdot}_{\varphi_i} &= \Omega R_i s_P^{(i)} \\
\Phi^{K\cdot}_{\varphi_j} &= -\Omega R_i s_P^{(j)}
\end{aligned}$$

Thus, in the jacobian matrix they will appear at the K^{th} coupple of lines and in the columns corresponding to $\begin{bmatrix} x_i & y_i & \varphi_i \end{bmatrix}$ and $\begin{bmatrix} x_j & y_j & \varphi_j \end{bmatrix}$, which means:

$$\Phi_q = \begin{bmatrix}
 & \cdots \begin{bmatrix} x_i & y_i & \varphi_i \end{bmatrix} & & \cdots \begin{bmatrix} x_j & y_j & \varphi_j \end{bmatrix} & \\
 & \vdots & & \vdots & \\
\cdots & \begin{bmatrix} \Phi^{K\cdot}_{r_i} & \Phi^{K\cdot}_{\varphi_i} \end{bmatrix}_{2\times 3} & \cdots & \begin{bmatrix} \Phi^{K\cdot}_{r_j} & \Phi^{K\cdot}_{\varphi_j} \end{bmatrix}_{2\times 3} & \cdots \\
 & \vdots & & \vdots &
\end{bmatrix} \begin{matrix} \\ \\ K_{2\times 1} \\ \\ \end{matrix}$$

For the rotational joints that connect a body j, the term $\begin{bmatrix} \Phi^{K\cdot}_{r_i} & \Phi^{K\cdot}_{\varphi_i} \end{bmatrix}$ will not be present in the Φ_q matrix.

If we look at an example in the code we will find:

```
%Rotation Joints
J( 1 :  2,  1 :  2 )= -eye(2);              % i=no   j=1    axis=o
J( 1 :  2,  3      )= -Omega* Rot1 *S1o;

J( 3 :  4, 19 : 20 )= -eye(2);              % i=no   j=7    axis=h
J( 3 :  4, 21      )= -Omega* Rot7 *S7h;

J( 5 :  6, 13 : 14 )= -eye(2);              % i=no   j=5    axis=n
J( 5 :  6, 15      )= -Omega* Rot5 *S5n;

J( 7 :  8,  1 :  2 )=  eye(2);              % i=1    j=2    axis=d
J( 7 :  8,  3      )=  Omega* Rot1 *S1d;
J( 7 :  8,  4 :  5 )= -eye(2);
J( 7 :  8,  6      )= -Omega* Rot2 *S2d;

J( 9 : 10,  1 :  2 )=  eye(2);              % i=1    j=3    axis=a
J( 9 : 10,  3      )=  Omega* Rot1 *S1a;
J( 9 : 10,  7 :  8 )= -eye(2);
J( 9 : 10,  9      )= -Omega* Rot3 *S3a;

J(11 : 12,  1 :  2 )=  eye(2);              % i=1    j=6    axis=m
J(11 : 12,  3      )=  Omega* Rot1 *S1m;
J(11 : 12, 16 : 17 )= -eye(2);
J(11 : 12, 18      )= -Omega* Rot6 *S6m;

J(13 : 14,  4 :  5 )=  eye(2);              % i=2    j=8    axis=g
J(13 : 14,  6      )=  Omega* Rot2 *S2g;
J(13 : 14, 22 : 23 )= -eye(2);
J(13 : 14, 24      )= -Omega* Rot8 *S8g;

J(15 : 16,  4 :  5 )=  eye(2);              % i=2    j=4    axis=c
J(15 : 16,  6      )=  Omega* Rot2 *S2c;
J(15 : 16, 10 : 11 )= -eye(2);
J(15 : 16, 12      )= -Omega* Rot4 *S4c;
```

```
35
36 J(17 : 18,  7 :  8 )=  eye(2);              % i=3    j=4    axis=b
37 J(17 : 18,  9      )=  Omega* Rot3 *S3b;
38 J(17 : 18, 10 : 11 )= -eye(2);
39 J(17 : 18, 12      )= -Omega* Rot4 *S4b;
```

For the K^{th} prismatic joints, between bodies i and j, the element of the jacobian matrix can be written as:

$$\begin{aligned}
\Phi_{q_i}^{K\angle} &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \\
\Phi_{q_j}^{K\angle} &= \begin{bmatrix} 0 & 0 & -1 \end{bmatrix} \\
\Phi_{q_i}^{K\uparrow} &= \begin{bmatrix} -\left(R_j v^{(j)}\right)^T & -\left(R_j v^{(j)}\right)^T \cdot \Omega R_i s_A^{(i)} \end{bmatrix} \\
\Phi_{q_j}^{K\uparrow} &= \begin{bmatrix} \left(R_j v^{(j)}\right)^T & -\left(R_j v^{(j)}\right)^T \cdot \Omega \cdot \left(r_j - r_i - R_i s_A^{(i)}\right) \end{bmatrix}
\end{aligned}$$

Thus, in the jacobian matrix they will appear at the K^{th} couple of lines (starting counting from the last line of rotational joint) and in the columns corresponding to $\begin{bmatrix} x_i & y_i & \varphi_i \end{bmatrix}$ and $\begin{bmatrix} x_j & y_j & \varphi_j \end{bmatrix}$, which means:

$$\Phi_q = \begin{bmatrix}
 & \cdots \begin{bmatrix} x_i & y_i & \varphi_i \end{bmatrix} & & \cdots \begin{bmatrix} x_j & y_j & \varphi_j \end{bmatrix} & \cdots & \\
 & \vdots & & \vdots & & \\
\cdots & \begin{bmatrix} \Phi_{q_i}^{K\angle} \\ \Phi_{q_i}^{K\uparrow} \end{bmatrix}_{2\times 3} & \cdots & \begin{bmatrix} \Phi_{q_j}^{K\angle} \\ \Phi_{q_j}^{K\uparrow} \end{bmatrix}_{2\times 3} & \cdots & K_{2\times 1} \\
 & \vdots & & \vdots & &
\end{bmatrix}$$

If we look at the code we will find:

```
1 % Prismatic Joints
2 J( 19, 15 ) = -1;
3 J( 20, 13 : 15 ) = [ (Rot5*v56)',  -(Rot5*v56)'*Omega*( F5(1:2) -F6(1:2) -Rot6*S6m )];
4 J( 19, 18 ) = 1;
5 J( 20, 16 : 18 ) = [-(Rot5*v56)',  -(Rot5*v56)'*Omega*Rot6*S6m];
6 J( 21, 21 ) = -1;
7 J( 22, 19 : 21 ) = [ (Rot7*v78)',  -(Rot7*v78)'*Omega*( F7(1:2) -F8(1:2) -Rot8*S8g )];
8 J( 21, 24 ) = 1;
9 J( 22, 22 : 24 ) = [-(Rot7*v78)',  -(Rot7*v78)'*Omega*Rot8*S8g ];
```

Finally the remaining lines of the jacobian matrix will be filled by the driving contraints. For the D^{th} driving constraint, between bodies i and j, the element of the jacobian matrix can be written as:

$$\begin{aligned}
\Phi_{q_i}^{D\uparrow} &= \begin{bmatrix} -\left(R_j u^{(j)}\right)^T & -\left(R_j u^{(j)}\right)^T \cdot \Omega R_i s_A \end{bmatrix} \\
\Phi_{q_j}^{D\uparrow} &= \begin{bmatrix} \left(R_j u^{(j)}\right)^T & -\left(R_j u^{(j)}\right)^T \cdot \Omega \cdot \left(r_j - r_i - R_i s_A^{(i)}\right) \end{bmatrix}
\end{aligned}$$

Thus, in the jacobian matrix they will appear at the D^{th} line (starting counting from the last line filled by prismatic joints joint) and in the columns corresponding to $\begin{bmatrix} x_i & y_i & \varphi_i \end{bmatrix}$ and $\begin{bmatrix} x_j & y_j & \varphi_j \end{bmatrix}$, which means:

$$\Phi_q = \begin{bmatrix} \cdots & \begin{bmatrix} x_i & y_i & \varphi_i \end{bmatrix} & \cdots & \begin{bmatrix} x_j & y_j & \varphi_j \end{bmatrix} & \cdots \\ \cdots & \begin{bmatrix} \Phi_{q_i}^{D\uparrow} \end{bmatrix}_{1\times 3} & \cdots & \begin{bmatrix} \Phi_{q_j}^{D\uparrow} \end{bmatrix}_{1\times 3} & \cdots \end{bmatrix} \begin{matrix} \\ D_{1\times 1} \end{matrix}$$

If we look at the code we will find:

```
%driving eq
 J( 23, 13 : 15 ) = [ (Rot5*u56)',  -(Rot5*u56)'*Omega*( F5(1:2) -F6(1:2) -Rot6*S6m )];
 J( 23, 16 : 18 ) = [-(Rot5*u56)',  -(Rot5*u56)'*Omega*Rot5*S6m];
 J( 24, 19 : 21 ) = [ (Rot7*u78)',  -(Rot7*u78)'*Omega*( F7(1:2) -F8(1:2) -Rot8*S8g )];
 J( 24, 22 : 24 ) = [-(Rot7*u78)',  -(Rot7*u78)'*Omega*Rot8*S8g];
```

The vector Φ is also a function of time and thus shall be derived with respect of time, but for practical purposes it is convenient to define a vector Φ_t that is separated from the matrix Φ_q.
If we look at the Φ vector we can notice that all constraints' equation non-related to driving constraints have no explicit expression of time. Thus, for all joints we can write:

$$\begin{aligned} \Phi_t^{K\cdot} &= 0_{2\times 1} \\ \Phi_t^{K\angle} &= 0_{1\times 1} \\ \Phi_t^{K\uparrow} &= 0_{1\times 1} \end{aligned}$$

the remaining lines of the Φ vector will be filled by the driving contraints. For the D^{th} driving constraint, between bodies i and j, the element of the jacobian matrix can be written as:

$$\Phi_t^{D\uparrow} = \frac{df(t)}{dt}$$

Thus, in the Φ_t vector it will appear at the D^{th} line (starting counting from the last line of prismatic joints).

$$\Phi_t = \begin{bmatrix} \\ \Phi_t^{D\uparrow} \\ \\ \end{bmatrix}_{1\times 1} \begin{matrix} \\ D_{1\times 1} \\ \\ \end{matrix}$$

If we look at the code we will find:

```
% Joints' equations
Ft=zeros(size(q));
% Driving equations
Ft(end-1)=     cos(2*t) /2;
Ft(end  )= -(3*cos(5*t))/2;
```

3.5. Γ Vector

The Γ vector represents

For the K^{th} prismatic joints, between bodies i and j and axis passing trough point P, the element of the Γ vector can be written as:

$$\Gamma^{K\cdot} = R_i s_P^{(i)} \dot{\varphi_i}^2 - R_j s_P^{(j)} \dot{\varphi_j}^2$$

For the rotational joints that connect a body j we will have $R_i s_P^{(i)} \dot{\varphi_i}^2 = 0$ In the Γ vector it will appear at the D^{th} line (starting counting from the last line of prismatic joints).

$$\Gamma = \begin{bmatrix} \vdots \\ \Gamma^{K\cdot}_{2\times 1} \\ \vdots \end{bmatrix} \quad \begin{matrix} \vdots \\ K_{2\times 1} \\ \vdots \end{matrix}$$

If we look at the code we will find:

```
% Rotational Joints
%G(k:k+1)=   Ri *Sia *(Fid(3)^2) -   Rj *Sja *(Fjd(3)^2)
G( 1:  2)= Rot0 *S0o *(F0d(3)^2) - Rot1 *S1o *(F1d(3)^2);      % i=0   j=1   a=o
G( 3:  4)= Rot0 *S0h *(F0d(3)^2) - Rot7 *S7h *(F7d(3)^2);      % i=0   j=7   a=h
G( 5:  6)= Rot0 *S0n *(F0d(3)^2) - Rot5 *S5n *(F5d(3)^2);      % i=0   j=5   a=n
G( 7:  8)= Rot1 *S1d *(F1d(3)^2) - Rot2 *S2d *(F2d(3)^2);      % i=1   j=2   a=d
G( 9: 10)= Rot1 *S1a *(F1d(3)^2) - Rot3 *S3a *(F3d(3)^2);      % i=1   j=3   a=a
G(11: 12)= Rot1 *S1m *(F1d(3)^2) - Rot6 *S6m *(F6d(3)^2);      % i=1   j=6   a=m
G(13: 14)= Rot2 *S2g *(F2d(3)^2) - Rot8 *S8g *(F8d(3)^2);      % i=2   j=8   a=g
G(15: 16)= Rot2 *S2c *(F2d(3)^2) - Rot4 *S4c *(F4d(3)^2);      % i=2   j=4   a=c
G(17: 18)= Rot3 *S3b *(F3d(3)^2) - Rot4 *S4b *(F4d(3)^2);      % i=3   j=4   a=b
```

For the K^{th} prismatic joints, between bodies i and j, the element of the Γ vector can be written as:

$$\begin{aligned} \Gamma^{K\angle} &= 0 \\ \Gamma^{K\uparrow} &= \left(R_j v^{(j)}\right)^T \cdot \left(2\Omega \cdot \left(\dot{r_j} - \dot{r_i}\right) \dot{\varphi_j} + \left(r_j - r_i\right) \dot{\varphi_j}^2 - R_i s_A^{(i)} \left(\dot{\varphi_j} - \dot{\varphi_i}\right)^2\right) \end{aligned}$$

Thus, in the Γ vector they will appear at the K^{th} couple of lines (starting counting from the last line of rotational joint).

$$\Gamma = \begin{bmatrix} \vdots \\ \begin{bmatrix} \Gamma^{K\angle} \\ \Gamma^{K\uparrow} \end{bmatrix}_{2\times 1} \\ \vdots \end{bmatrix} \quad \begin{matrix} \vdots \\ K_{2\times 1} \\ \vdots \end{matrix}$$

If we look at the code we will find:

```
% Prismatic Joints
G(20)= (Rot5*v56)' * (2*Omega*(F5d(1:2)-F6d(1:2))*F5d(3) + (F5(1:2)-F6(1:2))*(F5d(3)^2)...
            -Rot6*S6m*( (F5d(3)-F6d(3)) ^2) );
G(22)= (Rot7*v78)' * (2*Omega*(F7d(1:2)-F8d(1:2))*F7d(3) + (F7(1:2)-F8(1:2))*(F7d(3)^2)...
            -Rot8*S8g*( (F7d(3)-F8d(3)) ^2) );
```

Finally the remaining lines of the jacobian matrix will be filled by the driving contraints. For the D^{th} driving constraint, between bodies i and j, the element of the jacobian matrix can be written as:

$$\Gamma^{D\uparrow} = \left(R_j u^{(j)}\right)^T \cdot \left(2\Omega \cdot \left(\dot{r_j} - \dot{r_i}\right) \dot{\varphi_j} + \left(r_j - r_i\right) \dot{\varphi_j}^2 - R_i s_A^{(i)} \left(\dot{\varphi_j} - \dot{\varphi_i}\right)^2\right) - \frac{d^2 f(t)}{dt^2}$$

Thus, in the Γ vector it will appear at the D^{th} line (starting counting from the last line of prismatic joints).

$$\Gamma = \begin{bmatrix} \vdots \\ \Gamma^{D\uparrow}_{1\times 1} \\ \vdots \end{bmatrix} \quad \begin{matrix} \vdots \\ D_{1\times 1} \\ \vdots \end{matrix}$$

If we look at the code we will find:

```
% Driving Constraints
G(23)= (Rot5*u56)' * (2*Omega*(F5d(1:2)-F6d(1:2))*F5d(3) + (F5(1:2)-F6(1:2))*(F5d(3)^2)...
            -Rot6*S6m*( (F5d(3)-F6d(3)) ^2) ) + sin(2*t);
G(24)= (Rot7*u78)' * (2*Omega*(F7d(1:2)-F8d(1:2))*F7d(3) + (F7(1:2)-F8(1:2))*(F7d(3)^2)...
            -Rot8*S8g*( (F7d(3)-F8d(3)) ^2) ) - (15*sin(5*t))/2;
```

3.6. Newton-Raphson Method

Newton-Raphson Method is a method for finding successively better approximations to the zeroes of a real-valued function. The method starts with a function F defined over the real numbers x, the function's derivative F', and an initial guess x_0 for a root of the function f. If the function has a value lower than a given tolerance, we can consider to be satisfying the assumptions made and have found the roots of the function, else a better approximation x_1 is

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

For this project we can use q as the variable, the constraint vector Φ as the function for which we want to find the roots, and the jacobian matrix Φ_q as the function's derivative. Thus we get:

$$q_1 = q_0 - \Phi_q^{-1} \cdot \Phi$$

If we look at the code we will find an extra condition on the maximum number of iterations available to find the a satisfactory approximation of the roots:

```
while( (norm(F)>Tollerance) && (iter<max_iter) )
        F=Constratints(q,t);
        Fq=Jacobian_q(q);
        q=q - Fq\F; % it's equivalent to q=q-inv(Fq)*F
        iter=iter+1;
    end
```

3.7. Solving kinematics tasks

We now have all the elements necessary to solve position, velocity and acceleration problems. To do so we initialize the configuration vector to match the initial configuration of the system. Differently, its first and second derivative are set to have all elements equal to zero. At each instant of time we update the vectors as follow:

1. estimate the value of the current q using second order Taylor series:

$$q_{est} = q_{prev} + \dot{q}_{prev} + \frac{1}{2}\ddot{q}_{prev}$$

2. Starting from q_{est} and using Newton-Raphson method obtain a satisfactory value of q

3. compute the value of the velocity vector $\dot{q}$ as:

$$\dot{q} = -\Phi_q^{-1} \cdot \Phi_t$$

4. compute the value of the acceleration vector $\ddot{q}$ as:

$$\ddot{q} = -\Phi_q^{-1} \cdot \Gamma$$

```
T=0:dt:t_end;
for t=T
    q0  =q+q_d*dt+0*0.5*q_dd*(dt^2);
    q   =NewtonRaphson(q0, t);
    q_d =compute_vel(q,    t);
    q_dd=compute_acc(q,q_d,t);
end

% where
function q_d=compute_vel(q,t)
    Fq=Jacobian_q(q);
    Ft=zeros(size(q));
    Ft(end-1)=      cos(2*t)/2;
    Ft(end  )= -(3*cos(5*t))/2;
    q_d=-Fq\Ft;
end

function q_dd=compute_acc(q,q_d,t)
    Fq=Jacobian_q(q);
    Fg=Gamma(q,q_d,t);
    q_dd=Fq\Fg;
end
```

3.8. Visualization

For this model it has been created a visual output displaying the bodies, the characteristic points, the local frame origins and x axis orientation. The script also return an animation and a video of the moving system.

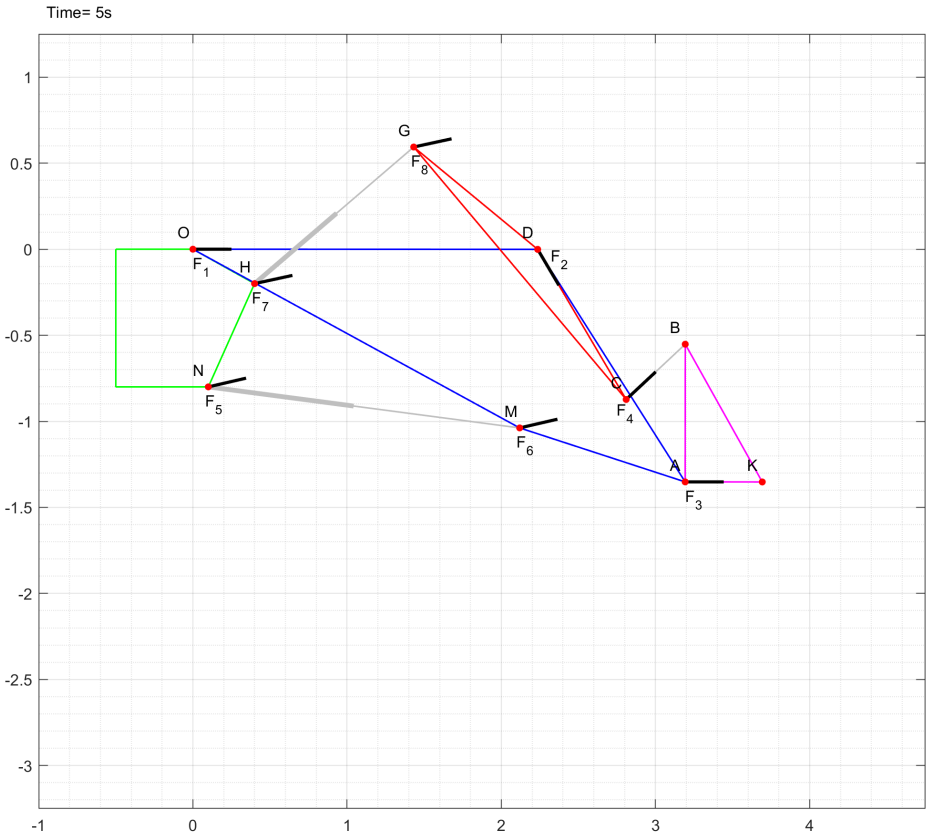

Figure 4: MATLAB model of the mechanism.

4. Comparison

The MATLAB model has been then compared to the ADAMS model over position, velocity and acceleration of several characteristic points. As visible in fig.5 position and velocity errors satisfactorily lay within $\pm 5 \cdot 10^{-7}$m. Regarding acceleration only one point (C) has values above $\pm 5 \cdot 10^{-7}$m but still withing the acceptable range $\pm 5 \cdot 10^{-6}$m. For those errors it is possible to compute standard deviation on all measurements, the result are visible in table 4.

Conclusions

For this paper it has been developed the kinematic model using both ADAMS and MATLAB for a givem mechnaism. Both model resulted to be correctly working, in particoular the MATLAB model has been verified through comparison with the ADAMS model over position, velocity and acceleration of several of the characteristic points.

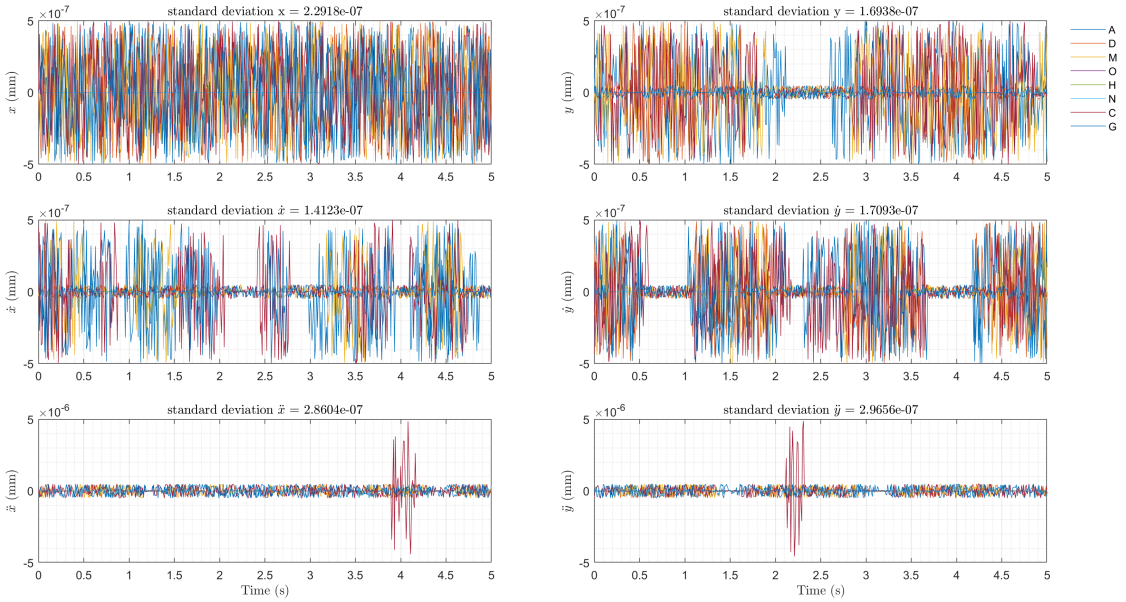

Figure 5: Errors between MATLAB and ADAMS models.

	std_x	std_y	$\text{std}_{\dot{x}}$	$\text{std}_{\dot{y}}$	$\text{std}_{\ddot{x}}$	$\text{std}_{\ddot{y}}$
Overall	2.2918e-07	1.6938e-07	1.4123e-07	1.7093e-07	2.8604e-07	2.9656e-07
A	2.9425e-07	2.7715e-07	2.2619e-07	2.3793e-07	2.6396e-07	2.7363e-07
D	2.9447e-07	1.581 e-07	2.3811e-08	2.1401e-07	1.8549e-07	2.3666e-07
M	2.9269e-07	2.5108e-07	1.7802e-07	1.9428e-07	2.3326e-07	2.5555e-07
O	5.2046e-12	5.523 e-12	7.0299e-17	1.1667e-16	1.9678e-16	2.1889e-16
H	0	0	3.8043e-17	5.3858e-17	1.7394e-16	2.2767e-16
N	0	0	1.5454e-16	1.2786e-16	3.7349e-16	3.0032e-16
C	2.8811e-07	2.5226e-07	2.083e-07	2.4153e-07	6.4536e-07	6.6173e-07
G	2.7954e-07	2.6348e-08	1.8009e-07	1.871e-07	2.8142e-07	2.6488e-07

Table 4: Standard deviation of the errors over results between MATLAB and ADAMS models.