Advanced Robotics Homework

Ermano Ardiles Arruda - 1391571

I. Introduction

This report outlines how the questions in the Advanced Robotics Homework were solved, and what results were achieved when tested on some instances of inputs. The version of the robot toolbox used in this homework was 9.8. Also, some experiments were performed as requested by some of the questions, and their results were analyzed.

II. Solutions

A. Question 1

Question 1 asks for the solution of forward kinematics problem given a table containing the DH parameters for each joint, along with a flag indicating its type (revolute or prismatic) and the vector representing the current configuration of the joints, q. It also asks for the computation of the Jacobian matrix, which describes the relationship between joint velocities $\dot{q}$ and end-effector velocities $\dot{p}_e$. Thus, this question asks for the implementation of the following function:

```
function [T,J] = fk(DH,q)
```

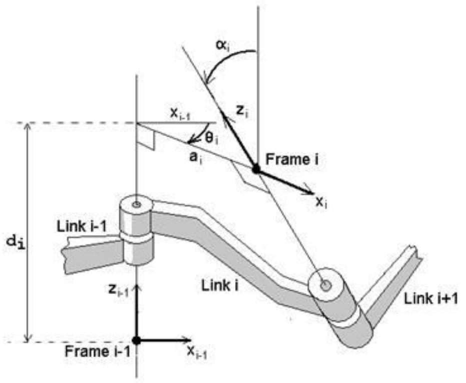
The function fk was then implemented according to the algorithms described in Siciliano's book as follows.

Once agreed to the Denavit-Hartenberg convention, given the DH parameters, the coordinate transformation between two frames $i-1$ and i is then described by the following two homogeneous transformations:

$$A_{i'}^{i-1} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_i^{i'} = \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

These transformations can be visually seen in the figure bellow, in which first the Frame i-1 is rotated around the z-axis by θ_i, and then translated by d_i in the z direction, i.e. $A_{i'}^{i-1}$, resulting in Frame i'. Finally, Frame i' is rotated around the x-axis by α_i, and then translated by a_i in the x direction, resulting in Frame i.



Hence,

$$A_i^{i-1} = A_{i'}^{i-1} A_i^{i'}$$

This is generalized to an open chain manipulator constituted by n+1 links as:

$$T_e^0 = T_0^b A_1^0 \dots A_n^{n-1} T_e^n$$

Which gives the position and orientation of the end-effector with respect to a base frame b. Function fk is a direct implementation of the computation of T_e^0 assuming that T_0^b and T_e^n are identity matrices, i.e. base frame corresponds to A_1^0 and end effector frame corresponds to A_n^{n-1}.

The second transformation computed by fk is the Jacobian. The Jacobian defines a linear transformation which represents how much the joints motion contributes to the final end effector linear and angular velocities. This relationship is given by:

$$v_e = \dot{p_e} = J\dot{q}$$

A fairly straightforward algorithm can be derived and executed right after the computation of the transformation T_e^0. This algorithm incrementally computes each column of the Jacobian matrix as follows:

$$\begin{bmatrix} J_{p_i} \\ J_{o_i} \end{bmatrix} = \begin{cases} \begin{matrix} z_{i-1} \\ \vec{0} \end{matrix} & for\ a\ prismatic\ joint \\ \begin{matrix} z_{i-1} \times (p_e - p_{i-1}) \\ z_{i-1} \end{matrix} & for\ a\ revolute\ joint \end{cases}$$

All required parameters can be extracted from the set of transformations used to compute T_e^0. Then, in the implementation of fk, the computation of the Jacobian was added as a subsequent step.

In the code submitted which contains the file fk.m, there's also another source file called fk_test.m which performs a series of tests with the computed T_e^0 and Jacobian J transformations, given a default robot arm and a set of random generated configurations q. For each T_e^0 and Jacobian J computed by the function fk, a corresponding T'^0_e and Jacobian J' is computed using the robot toolbox for the same robot with the same joint configuration.

These corresponding transformations should be the same, for the sake of correctness. Then, they are compared, taking into consideration eventual floating point precision problems, and tested to see whether they are equal or not. The implementation showed to be correct and bug free after the execution of such a test.

B. *Question 2*

In question 2, the task was to solve the planar version of the inverse kinematics problem with an iterative solution showed in lecture and also described in Siciliano's Book with several variations. Thus, given the table with the DH parameters of a multi-link robot arm, a current configuration q, the main and secondary goals, namely the desired end effector position and orientation p_des and configuration q_{des}, respectively. The task was to compute q_{new}, the final configuration of the robot arm, such that it primarily maintains $p_{des} - p_e = 0$, and secondarily minimizes $q_{des} - q$. The flag Mq indicates if the secondary goal must be pursued or not.

```
function q_new = ik(DH,q,p_des,q_des,Mq)
```

More precisely, in the submitted implementation, q_new is a matrix where each row is a configuration q(t), part of the trajectory of the robot arm starting in q(0)=q and ending in q(f)=qf, which is the final configuration to achieve p_des and its secondary goal q_des (if Mq is set 1). In other words, q_new contains the intermediary results of the iterative integration process. The matrix q_new follows the robot toolbox convetion for ploting the trajectory of the robot towards the final end effector pose. Therefore, this choice was made for visualization purposes. The iterative integration process is described by the equation bellow and basically defines the algorithm for the ik function.

$$q(t+1) = q(t) + \dot{q}(t)\Delta t$$

Where,

$$\dot{q}(t) = J^{*}Ke + (I_n - J^{*}J)\dot{q}_0$$

$$e = p_{des} - p_e$$

$$\dot{q}_0 = q_{des} - q$$

And J^{*} is the damped least-squares (DLS) inverse, which is defined as:

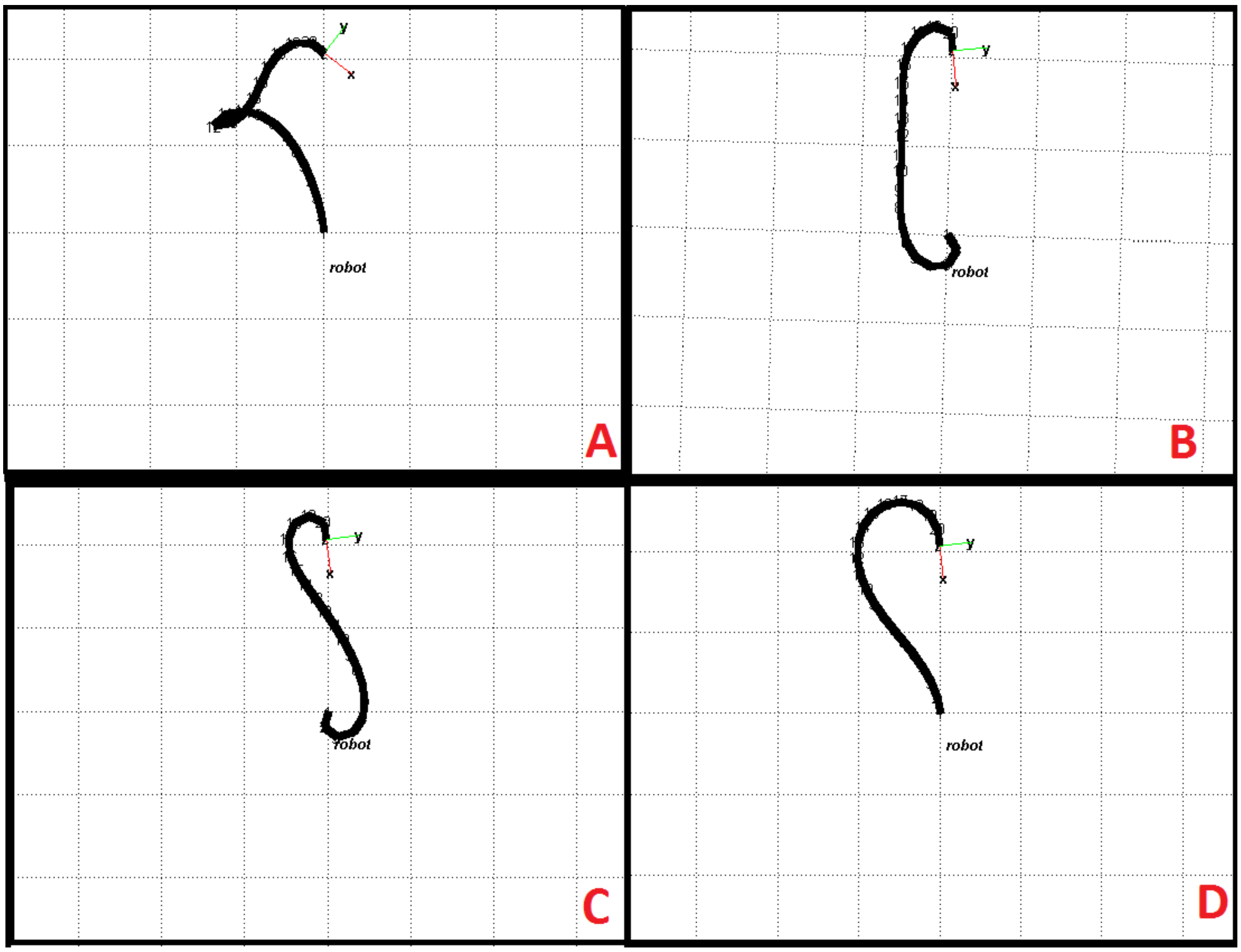
$$J^{*} = J^{T}(JJ^{T} + k^{2}I)^{-1}$$

The DLS inverse was chosen to avoid singularity problems. This way testing was made easier, for it wasn't necessary to worry about eventual singular configurations the robot could assume. The value for the constant damping factor k was arbitrarily chosen to be 0.3. Although in real life scenarios this would be subject of experimentation, this value worked fine during simulations and, therefore, was kept. Also, the matrix K was set to identity and the integration time Δt was set to 0.05 seconds. This way, $q(t)$ is updated at each iteration t, yielding $q(t+1)$. The terminating condition was defined in a way that it's set to true if the magnitude of the error e is less than a threshold (Error_threshold = 0.5) or if a limit of iterations (Nit) was reached. By adopting this combined terminating condition, we make sure the error of e gets close to zero, and also cover the cases where it's not possible to reach p_{des}, which would lead to an infinite loop if the terminating condition didn't include a limit for iterations.

An auxiliary function called default_dh was defined. This function creates a table of DH parameters for a 20 link robot arm. This robot arm was used to test the function ik. One of the tasks consisted of drawing letters on the plot surface by assuming particular configurations. Bellow, image A depicts a lower case R, image B depicts a letter C, image C depicts a letter S, and image D depicts what could be seen as a half-heart. All the images have the same $p_{des} = [0\ 10\ 0\ 0\ 0 - \frac{\pi}{4}]^T$, only the q_{des} parameter varied in order to draw the different letters, the different values for q_{des} were (in matlab notation):

- `For drawing an R: [ pi/2; zeros(10,1); -pi/2; zeros(8,1)]`
- `For drawing a C: [ pi; zeros(19,1)]`
- `For drawing an S: [ 0; zeros(19,1)]`

In order to draw the half-heart, the flag Mq was set to 0, meaning that the secondary goal was not pursued.



Finally, to demonstrate that the robot tries as hard as it can to assume the secondary goal, which is the final desired configuration q_{des}, an experiment was performed with a 3 link robot arm defined by its DH parameters in the table DH. Then, q_{des} was set in such way that its first link is desired to make an angle of 0° with the x-axis, the second link is desired to make an angle of 90° degrees with the first link (consequently, with the x-axis) and the third, an angle of 0° with the second link. The initial configuration q was defined as the null-vector. Lastly, the desired end effector position and orientation was set to be $p_{des} = [\,2\;2\;0\;0\;0\,\frac{\pi}{2}]^T$, which is the point (2,2) in the plane, with orientation around the z-axis of 90°.

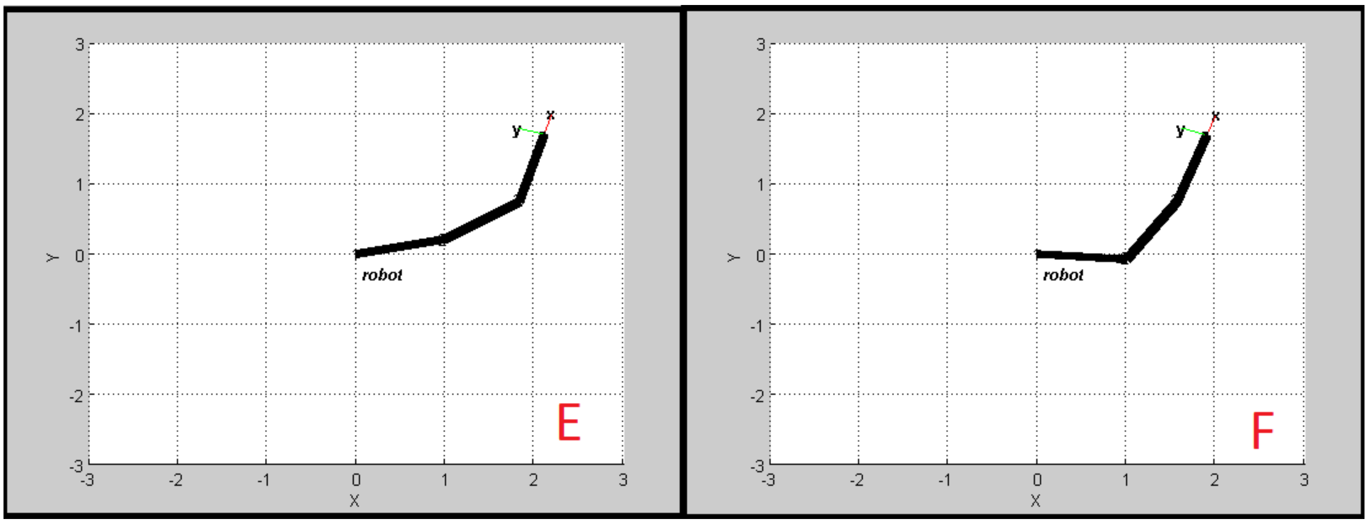
$$q_{des} = [0\,\frac{\pi}{2}\,0]^T$$

$$DH = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$q = [0\;0\;0]^T$$

$$p_{des} = [\,2\;2\;0\;0\;0\,\frac{\pi}{2}]^T$$

This setup was tested with Mq = 0, and Mq = 1, respectively. The results obtained can be seen bellow:



It can be seen above that the final end effector position is slightly different between E and F due to the adopted threshold of 0.5. Also, it’s likely that, due to numerical problems, e.g. floating point precision problems, the null space projection does generates pure self-motion without slightly affecting the end effector pose. However, this hypothesis was not tested, and the subtle imprecision to achieve the final goal is not an issue in this application. Figure E is the final posture of the robot with Mq = 0, whereas figure F is the final posture of the robot with Mq = 1, i.e. considering the secondary goal. We can clearly see that the algorithm did what it could to make the second link make an angle as close as possible to 90°. The robot doesn’t have enough degrees of freedom to achieve a perfect match with q_{des} , but it gets close enough, while respecting the error threshold of 0.5 and achieving the primary goal.

The code submitted as solution has other useful functions that can be used for testing, they are:

```
function robota = constructRobot(DH)
```

Which receives a table with the DH parameters for each joint, plus one extra column for joint type specification. The return is a SerialLink robot from the robot toolbox, which can be used for ploting.

```
function q_new = ik(DH,q,p_des,q_des,Mq)
```

This is the function solution for this question, q_new is a matrix where, which row is a configuration at time step t, as explained before. The result q_new can be directly used to plot and animate the robot in the screen by calling:

```
plot(r,q_new)
```

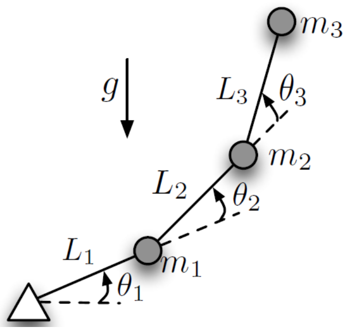
Where, r is a robot returned by the function constructRobot.

Finally, the function ik_test bellow can be called with no arguments in order to test the solution with some predefined inputs. Otherwise, it receives the same arguments as ik, and as a shortcut, it plots the result q_new on the screen for direct testing.

```
function ik_test(DH, q,p_dest, q_dest, Mq)
```

C. Question 3

About this question, in this report I answer letters a), b) and c) by hand. Letter d) was symbolically derived using Mathematica. The Mathematica files among with a pdf of the derivation can be viewed in the folder named "Appendices" submitted with this report. There are some scans of my original handmade transcripts for letters a), b) and c) at the end of this report. There's also a full symbolic derivation that answers letters a), b), c) and d), created on Mathematica without using my handmade derivation of the Lagragian L as a starting point. These pdf and Mathematica (.nb) files are also in the folder "Appendices". Unfortunately, letter e) was not answered. The handmade derivation is summarized below.



Trigonometric identities:

$$\sin(u)\sin(v) = \frac{1}{2}[\cos(u-v) - \cos(u+v)]$$

$$\cos(u)\cos(v) = \frac{1}{2}[\cos(u-v) + \cos(u+v)]$$

$$sin^2(u) + cos^2(u) = 1$$

a. Derive the Kinetic Energy of the system

$$KE = KE_1 + KE_2 + KE_3$$

Kinect Energy for Link 1 (KE_1)

$$KE_1 = \frac{1}{2} m_1 v_1^T v_1 + \frac{1}{2} \omega_1^T I_1 \omega_1$$

$$p_1 = \begin{bmatrix} L_1 cos(\theta_1) \\ L_1 sin(\theta_1) \\ 0 \end{bmatrix}$$

$$v_1 = \begin{bmatrix} -L_1 \dot{\theta}_1 \sin(\theta_1) \\ L_1 \dot{\theta}_1 sin(\theta_1) \\ 0 \end{bmatrix}$$

$$\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}, I_1 = \begin{bmatrix} Ixx_1 & 0 & 0 \\ 0 & Iyy_1 & 0 \\ 0 & 0 & Izz_1 \end{bmatrix}$$

$$KE_1 = \frac{1}{2} m_1 L_1^2 \dot{\theta}_1^2 + \frac{1}{2} Izz_1 \dot{\theta}_1^2$$

Kinect Energy for Link 2 (KE_2)

$$KE_2 = \frac{1}{2} m_2 v_2^T v_2 + \frac{1}{2} \omega_2^T I_2 \omega_2$$

$$p_2 = \begin{bmatrix} L_1 cos(\theta_1) + L_2 cos(\theta_1 + \theta_2) \\ L_1 sin(\theta_1) + L_2 sin(\theta_1 + \theta_2) \\ 0 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} -L_1 \dot{\theta}_1 sin(\theta_1) - L_2(\dot{\theta}_1 + \dot{\theta}_2) sin(\theta_1 + \theta_2) \\ L_1 \dot{\theta}_1 cos(\theta_1) + L_2(\dot{\theta}_1 + \dot{\theta}_2) cos(\theta_1 + \theta_2) \\ 0 \end{bmatrix}$$

$$\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix}, I_2 = \begin{bmatrix} Ixx_2 & 0 & 0 \\ 0 & Iyy_2 & 0 \\ 0 & 0 & Izz_2 \end{bmatrix}$$

$$KE_2 = \frac{1}{2} m_2 \left[L_1^2 \dot{\theta}_1^2 + L_2^2 \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 + 2 L_1 L_2 \dot{\theta}_1 \left(\dot{\theta}_1 + \dot{\theta}_2\right) \cos(\theta_2) \right] + \frac{1}{2} Izz_2 \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2$$

Kinect Energy for Link 3 (KE_3)

$$KE_3 = \frac{1}{2}m_3 v_3^T v_3 + \frac{1}{2}\omega_3^T I_3 \omega_3$$

$$p_3 = \begin{bmatrix} L_1cos(\theta_1) + L_2cos(\theta_1+\theta_2) + L_3cos(\theta_1+\theta_2+\theta_3) \\ L_1sin(\theta_1) + L_2sin(\theta_1+\theta_2) + L_3sin(\theta_1+\theta_2+\theta_3) \\ 0 \end{bmatrix}$$

$$v_3 = \begin{bmatrix} -L_1\dot{\theta}_1sin(\theta_1) - L_2(\dot{\theta}_1+\dot{\theta}_2)sin(\theta_1+\theta_2) - L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)sin(\theta_1+\theta_2+\theta_3) \\ L_1\dot{\theta}_1cos(\theta_1) + L_2(\dot{\theta}_1+\dot{\theta}_2)cos(\theta_1+\theta_2) + L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)cos(\theta_1+\theta_2+\theta_3) \\ 0 \end{bmatrix}$$

$$\omega_3 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3 \end{bmatrix}, I_3 = \begin{bmatrix} Ixx_3 & 0 & 0 \\ 0 & Iyy_3 & 0 \\ 0 & 0 & Izz_3 \end{bmatrix}$$

$$\begin{aligned} KE_3 = \frac{1}{2}m_3 \Big[& L_1^2\dot{\theta}_1^2 + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ & + 2(L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_2) + L_1L_3\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)\cos(\theta_2+\theta_3) + L_2L_3(\dot{\theta}_1 \\ & + \dot{\theta}_2)(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)\cos(\theta_3))\Big] + \frac{1}{2}Izz_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2 \end{aligned}$$

Kinect Energy of the System (Total Kinect Energy, KE)

$$KE = KE_1 + KE_2 + KE_3$$

$$\begin{aligned} KE = & \frac{1}{2}m_1L_1^2\dot{\theta}_1^2 + \frac{1}{2}Izz_1\dot{\theta}_1^2 + \frac{1}{2}m_2\Big[L_1^2\dot{\theta}_1^2 + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2 + 2L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_2)\Big] + \frac{1}{2}Izz_2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ & + \frac{1}{2}m_3\Big[L_1^2\dot{\theta}_1^2 + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ & + 2(L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_2) + L_1L_3\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)\cos(\theta_2+\theta_3) + L_2L_3(\dot{\theta}_1 \\ & + \dot{\theta}_2)(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)\cos(\theta_3))\Big] + \frac{1}{2}Izz_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2 \end{aligned}$$

b. Derive the Potential Energy of the system.

$$PE = PE_1 + PE_2 + PE_3$$

Potential Energy for Link 1 (PE_1)

$$PE_1 = m_1gL_1sin(\theta_1)$$

Potential Energy for Link 2 (PE_2)

$$PE_1 = m_2g(L_1sin(\theta_1) + L_2sin(\theta_1+\theta_2))$$

Potential Energy for Link 3 (PE_3)

$$PE_3 = m_2g(L_1sin(\theta_1) + L_2sin(\theta_1+\theta_2) + L_3sin(\theta_1+\theta_2+\theta_3))$$

Potential Energy of the System (Total Potential Energy, PE)

$$PE = PE_1 + PE_2 + PE_3$$

$$PE = m_1 g L_1 sin(\theta_1) + m_2 g(L_1 sin(\theta_1) + L_2 sin(\theta_1 + \theta_2)) + m_2 g(L_1 sin(\theta_1) + L_2 sin(\theta_1 + \theta_2) + L_3 sin(\theta_1 + \theta_2 + \theta_3))$$

c. Derive the Lagrangian, L, for the system.

Lagrangian (L)

$$L = KE - PE$$

$$L = \frac{1}{2} m_1 L_1^2 \dot{\theta}_1^2 + \frac{1}{2} Izz_1 \dot{\theta}_1^2 + \frac{1}{2} m_2 \left[L_1^2 \dot{\theta}_1^2 + L_2^2 (\dot{\theta}_1 + \dot{\theta}_2)^2 + 2 L_1 L_2 \dot{\theta}_1 (\dot{\theta}_1 + \dot{\theta}_2) \cos(\theta_2)\right] + \frac{1}{2} Izz_2 (\dot{\theta}_1 + \dot{\theta}_2)^2 + \frac{1}{2} m_3 \left[L_1^2 \dot{\theta}_1^2 + L_2^2 (\dot{\theta}_1 + \dot{\theta}_2)^2 + 2(L_1 L_2 \dot{\theta}_1 (\dot{\theta}_1 + \dot{\theta}_2) \cos(\theta_2) + L_1 L_3 \dot{\theta}_1 (\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3) \cos(\theta_2 + \theta_3) + L_2 L_3 (\dot{\theta}_1 + \dot{\theta}_2)(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3) \cos(\theta_3))\right] + \frac{1}{2} Izz_3 (\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3)^2 - [m_1 g L_1 sin(\theta_1) + m_2 g(L_1 sin(\theta_1) + L_2 sin(\theta_1 + \theta_2)) + m_2 g(L_1 sin(\theta_1) + L_2 sin(\theta_1 + \theta_2) + L_3 sin(\theta_1 + \theta_2 + \theta_3))]$$

d. Derive each term for the required partial derivatives individually.

This derivation is in the folder named "Appendices" submitted with this report.

D. *Question 4*

In question 4, the proposed task was to use the recursive Newton-Euler Algorithm to create a simulation of a planar robot with a series of N revolute links. To solve the task, the first step was to implement the Newton-Euler algorithm as the following function:

```
function [tau] = NewtonEuler(N,L,m,I,fv,fc,g,q,qd,qdd)
```

Where,

- *N* is the number of links.
- *L* is the length of the links (it was assumed each link has the same length).
- *m* is the mass of each link (it was assumed each link has the same mass).
- *I* is the moment of inertia of each link (it was assumed each link has the same moment of inertia).
- *fv* is the coefficient of viscous friction for each link (assumed the same for all links).
- *fc* is the coefficient of coulomb friction for each link (assumed the same for all links).
- *g* is the gravity vector with respect to the base frame. This vector was set to $[0\ \ -9.81\ 0]^T$.
- *q* is the configuration of the robot.
- *qd* is the velocity of the robot.
- *qdd* is the acceleration of the robot.

In summary, given the physical parameters of a given robot (*N, L, m, I, fv, fc, g*), its current joint configurations, velocities and accelerations (*q, qd, qdd),* the Newton-Euler Algorithm computes the final vector *tau,* which represents the final torques at the joints. In addition to these parameters, additional parameters for the rotor were defined, namely *Im*, *mm*, *kri*, which are, respectively, the moment of inertia, mass and gear reduction ratio of each rotor.

The Newton-Euler formulation enables the implementation of a recursive alternative for the Lagrange formulation of the system. It describes the motion of a link according to the balance of forces and moments acting on it [Siciliano]. This means that the Newton-Euler formulation is a different method for evaluating the following dynamic model for the inverse dynamics problem:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = \tau$$

Where,

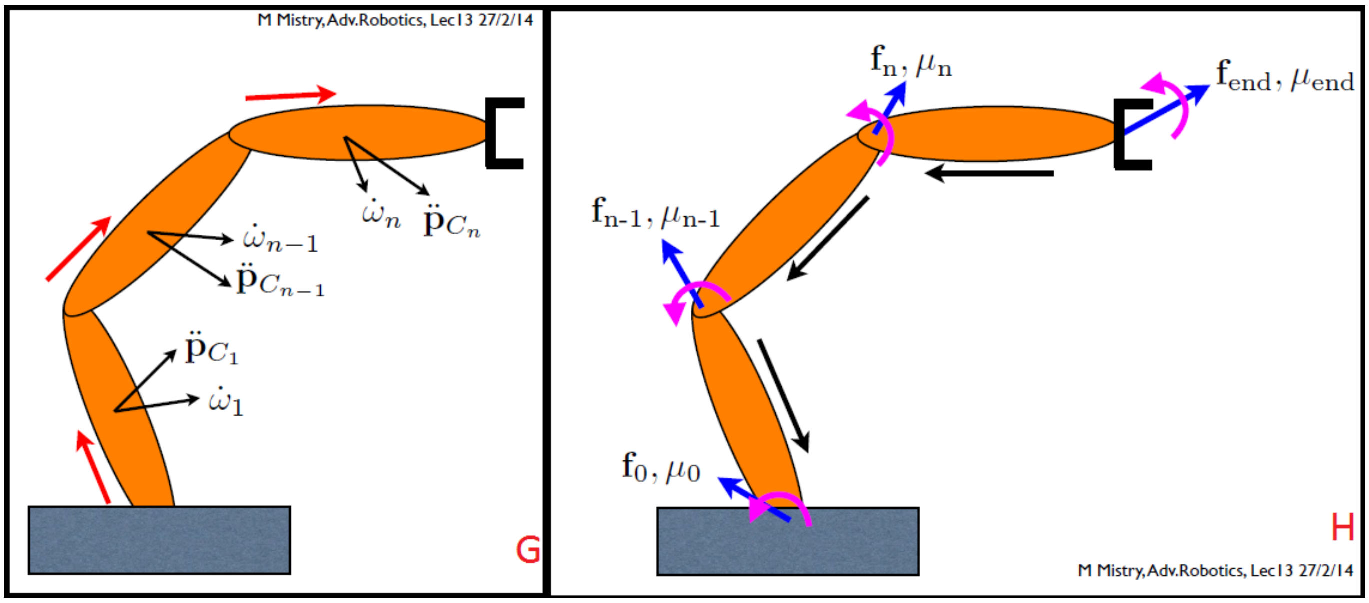
$M(q)$ is the inertia matrix of the system

$C(q,\dot{q})$ is composed of the Coriolis and Centrifugal components and

$G(q)$ is the gravity vector

The model evaluates to the vector τ (tau), the Nx1 vector representing the torques at each joints. The computation of tau given the current joint positions, velocities and accelerations is known as the *Inverse Dynamics* problem.

The Newton-Euler Algorithm is composed of two phases. First, a forward phase that propagates the velocities and accelerations from base to end-effector is carried out. Secondly, a backward phase is executed, which is responsible for propagating the forces and moments from end-effector to the base. These two phases are depicted in figures G and H, respectively.



Therefore, in the forward phase, for each link i in the open chain depicted in G, given $q, \dot{q}, \ddot{q}$ and also setting the terminals $\omega_0 = [0\ 0\ 0]^T$, $\ddot{p}_0 - g_0 = [0\ g\ 0]^T$, $\dot{\omega}_0 = [0\ 0\ 0]^T$, the vectors ω_i, $\dot{\omega}_i^i$, $\ddot{p}_i^i$, $\ddot{p}_{Ci}^i$ and $\dot{\omega}_{mi}^{i-1}$ are computed. These angular and linear velocities and accelerations are then used in the backward phase in order to compute the forces f_i^i and moments u_i^i, from end-effector to base link, once having set the terminals $f_{n+i}^{n+i} = u_{n+i}^{n+i} = [\ 0\ 0\ 0\ ]^T$. The equations for computing all the cited terms can be found in chapter 7 of Siciliano's book, and also in the submitted code, where the viscous and coulomb friction were also integrated to the model.

After having implemented the Newton-Euler algorithm, the second step is to simulate an arbitrary robot with N links under the effects of the physical formulation of the system. To do that, it's possible to isolate the acceleration term of the system by pre multiplying the model by the inverse of the inertia matrix, $M^{-1}(q)$. It's worth mentioning that $M(q)$ is positive definite and, therefore, is always invertible.

$$\ddot{q} = M^{-1}(q)(\tau_{act} - C(q, \dot{q}) - G(q))$$

$$\ddot{q} = M^{-1}(q)(\tau_{act} - \tau')$$

Where,

$$\tau' = C(q, \dot{q}) + G(q))$$

The computation of joint accelerations $\ddot{q}$ given the joint torques τ' is defined as the problem of *Forward Dynamics.*

Each column b_i of the inertia matrix $M(q)$ can be obtained by using the Newton-Euler algorithm as a black box, passing setting $q = \dot{q} = \vec{0}$, $\ddot{q}_i = 1$ and $\ddot{q}_j = 0$ for $i \neq j$. And to compute τ', one must set $\ddot{q} = \vec{0}$. Finally, at each time step t of the simulation, the following algorithm was implemented:

Algorithm `fdyn_ne`

```
Input: tau_act,N,L,m,I,fv,fc,g,q,qd,qdd
Output: trajectory q(t) of the robot arm
```

For each time step t:

1. Compute $M(q)$ as described above using the Newton-Euler algorithm as a black box
2. Compute $M^{-1}(q)$
3. Compute τ' as described above using the Newton-Euler algorithm as a black box
4. Compute $\ddot{q} = M^{-1}(q)(\tau_{act} - \tau')$
5. Using the Euler integration method with integration time Δt, compute $\dot{q} = \dot{q_0} + \Delta t\ddot{q}$
6. Using the Euler integration method with integration time Δt, compute $q = q_0 + \Delta t\dot{q}$
7. Store the current configuration of time step t, $q(t) = q$

Where, Δt was set to 0.5 seconds

In the submitted code, the code that executes this procedure is the matlab function:

```
function [robota,q_out] = fdyn_ne(N)
```

This function receives the number of links and executes the simulation described in this report for a given number of links *N*. This function also plots the computed trajectory of the robot arm, along with the animation of its movement.

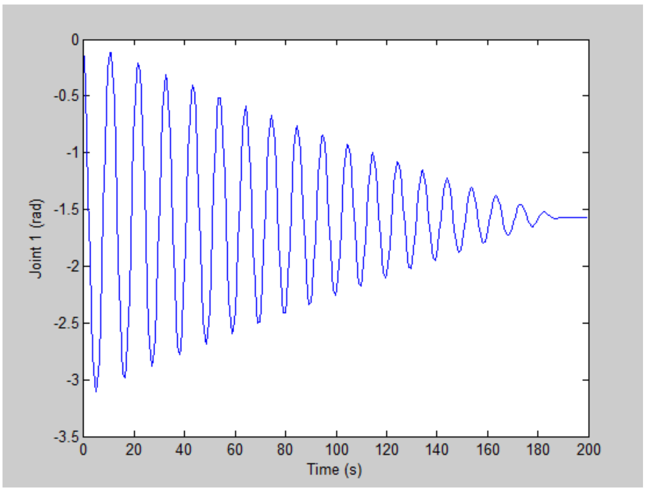
Having explained the general procedures for the simulation, two different robot arms were tested. The first was a 1-link robot arm (i.e. a pendulum), and the second was a 3-link robot arm. The following simulation parameters were set for both:

- $q_0 = \vec{0}$
- $\dot{q_0} = \vec{0}$
- $\tau_{act} = \vec{0}$
- $g = [0\ -9.81\ 0]'(m/s^2)$
- $fv = 0.05$
- $fc = 0.9$
- $I = 10\ kg \cdot m^2$
- $L = 2\ m$
- $m = 5\ kg$
- $kri = 100$
- $Im = 0.005\ kg \cdot m^2$
- $mm = 5\ kg$
- $\tau_{act} = \vec{0}$

As can be observed, the simulation is performed with zero actuator torques. Thus, this setup simulates the robot left for the action of gravity, given its initial configuration until it reaches an equilibrium state, which is having the robot arm stretched towards the gravity vector.

Simulation of 1 Link Robot Arm

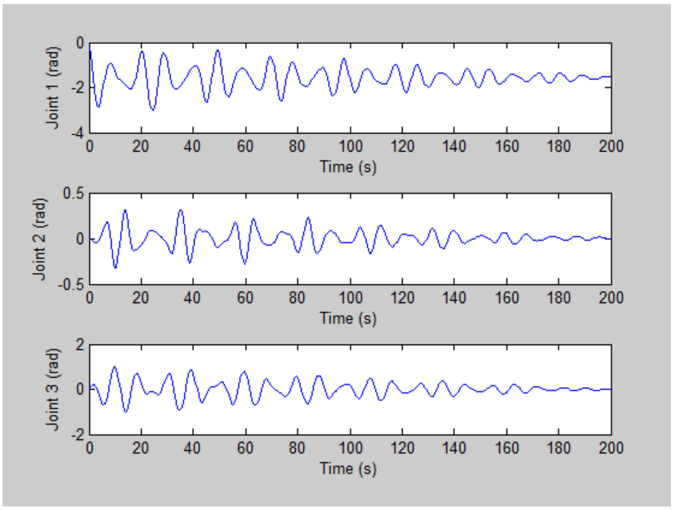
When the simulation was performed for the 1 link robot arm, the following trajectory for the single joint variable:



This trajectory over time behaves as expected, as the pendulum loses energy over time, until it reaches its equilibrium state, which is to be positioned with an angle of $-\pi/2$ with the x-axis, i.e. pointing towards the ground, or in the same direction as the gravity vector. A video of the simulation can be seen in the following link: https://www.youtube.com/watch?v=4sk41eEoY3I.

Simulation of 3 Link Robot Arm

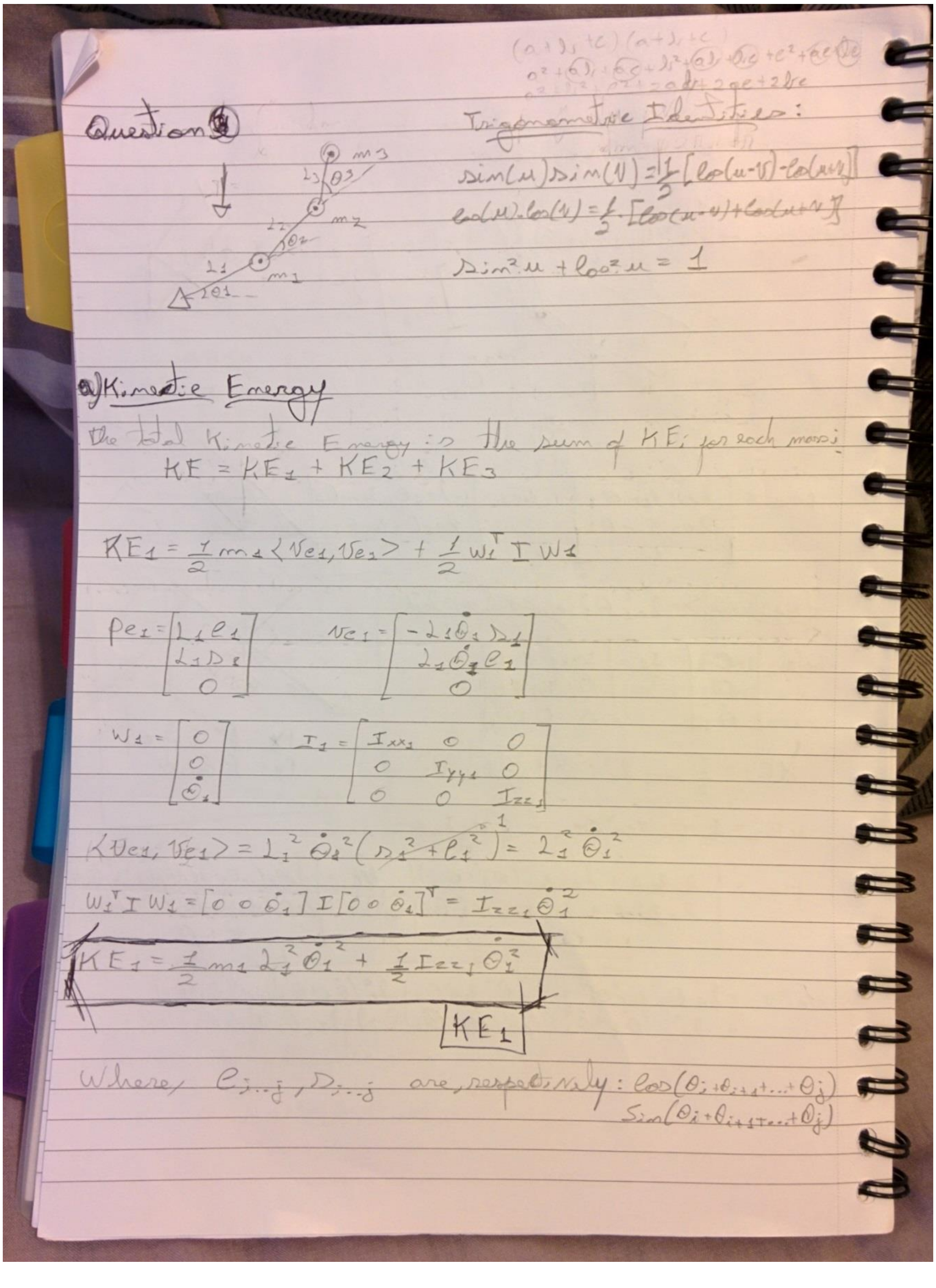
For the 3 link robot arm, the following trajectory was obtained for the three joint variables:



In a similar way, the 3 link robot behaves like a triple pendulum and, likewise, loses energy over time, assuming the same equilibrium state for each variable. Ultimately, this brings the robot to be with its arm stretched towards the same direction as the gravity vector. A video for this simulation was recorded, and can be seen on https://www.youtube.com/watch?v=xwTJKXfld7k.

The code submitted along with the registered results and background information written in this section concludes the simulation for Question 4.

E. Scans Questions 3

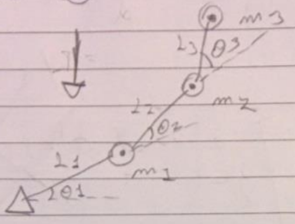


$(a+b+c)(a+b+c)$

$a^2+ab+ac+b^2+ab+bc+c^2+ac+bc$

$a^2+b^2+c^2+2ab+2ac+2bc$

Question [illegible]



Trigonometric Identities:

$\sin(u)\sin(v) = \frac{1}{2}[\cos(u-v) - \cos(u+v)]$

$\cos(u).\cos(v) = \frac{1}{2}.[\cos(u-v) + \cos(u+v)]$

$\sin^2 u + \cos^2 u = 1$

a) Kinetic Energy

The total Kinetic Energy is the sum of KE_i for each mass:

$$KE = KE_1 + KE_2 + KE_3$$

$$KE_1 = \frac{1}{2} m_1 \langle v_{e1}, v_{e1} \rangle + \frac{1}{2} w_1^T I w_1$$

$$p_{e1} = \begin{bmatrix} L_1 c_1 \\ L_1 s_1 \\ 0 \end{bmatrix} \qquad v_{e1} = \begin{bmatrix} -L_1 \dot{\theta}_1 s_1 \\ L_1 \dot{\theta}_1 c_1 \\ 0 \end{bmatrix}$$

$$w_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \qquad I_1 = \begin{bmatrix} I_{xx_1} & 0 & 0 \\ 0 & I_{yy_1} & 0 \\ 0 & 0 & I_{zz_1} \end{bmatrix}$$

$$\langle v_{e1}, v_{e1} \rangle = L_1^2 \dot{\theta}_1^2 \overset{1}{\cancel{(s_1^2 + c_1^2)}} = L_1^2 \dot{\theta}_1^2$$

$$w_1^T I w_1 = [0 \ 0 \ \dot{\theta}_1] \, I \, [0 \ 0 \ \dot{\theta}_1]^T = I_{zz_1} \dot{\theta}_1^2$$

$$\boxed{KE_1 = \frac{1}{2} m_1 L_1^2 \dot{\theta}_1^2 + \frac{1}{2} I_{zz_1} \dot{\theta}_1^2}$$

$\boxed{KE_1}$

Where, $c_{i..j}$, $s_{i..j}$ are, respectively: $\cos(\theta_i + \theta_{i+1} + \ldots + \theta_j)$
$\sin(\theta_i + \theta_{i+1} + \ldots + \theta_j)$

$(x+y)^2 = x^2+2xy$

$$KE_2 = \frac{1}{2} m_2 \langle v_{e_2}, v_{e_2} \rangle + \frac{1}{2} \omega_2^T I \omega_2$$

$$p_{e_2} = \begin{bmatrix} L_1 c_1 + L_2 c_{12} \\ L_1 s_1 + L_2 s_{12} \\ 0 \end{bmatrix} \qquad v_{e_2} = \begin{bmatrix} -L_1 \dot\theta_1 s_1 - L_2(\dot\theta_1+\dot\theta_2) s_{12} \\ L_1 \dot\theta_1 c_1 + L_2(\dot\theta_1+\dot\theta_2) c_{12} \\ 0 \end{bmatrix}$$

$$\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot\theta_1+\dot\theta_2 \end{bmatrix} \qquad I_2 = \begin{bmatrix} I_{xx_2} & 0 & 0 \\ 0 & I_{yy_2} & 0 \\ 0 & 0 & I_{zz_2} \end{bmatrix}$$

$$\begin{aligned}\langle v_{e_2}, v_{e_2} \rangle &= L_1^2 \dot\theta_1^2 s_1^2 + 2L_1L_2\dot\theta_1(\dot\theta_1+\dot\theta_2) s_1 s_{12} + L_2^2(\dot\theta_1+\dot\theta_2)^2 s_{12}^2 \\ &\quad + L_1^2\dot\theta_1^2 c_1^2 + 2L_1L_2\dot\theta_1(\dot\theta_1+\dot\theta_2).c_1c_{12} + L_2^2(\dot\theta_1+\dot\theta_2)^2 c_{12}^2 \\ &= L_1^2\dot\theta_1^2(s_1^2+c_1^2) + 2L_1L_2\dot\theta_1(\dot\theta_1+\dot\theta_2).(s_1s_{12}+c_1c_{12}) \\ &\quad + L_2^2(\dot\theta_1+\dot\theta_2)^2(s_{12}^2 + c_{12}^2)\end{aligned}$$

$$\langle v_{e_2}, v_{e_2} \rangle = L_1^2\dot\theta_1^2 + L_2^2(\dot\theta_1+\dot\theta_2)^2 + 2L_1L_2\dot\theta_1(\dot\theta_1+\dot\theta_2)\underbrace{(s_1s_{12}+c_1c_{12})}_{c_2}$$

$$\langle v_{e_2}, v_{e_2} \rangle = L_1^2\dot\theta_1^2 + L_2^2(\dot\theta_1+\dot\theta_2)^2 + 2L_1L_2\dot\theta_1(\dot\theta_1+\dot\theta_2) c_2$$

$$\boxed{KE_2 = \frac{1}{2} m_2 \left[L_1^2\dot\theta_1^2 + L_2^2(\dot\theta_1+\dot\theta_2)^2 + 2L_1L_2\dot\theta_1(\dot\theta_1+\dot\theta_2)c_2\right] + \frac{1}{2} I_{zz_2}(\dot\theta_1+\dot\theta_2)^2}$$

$\boxed{KE_2}$

$$KE_3 = \frac{1}{2} m_3 \langle v_{e_3}, v_{e_3} \rangle + \frac{1}{2} \omega_3^T I \omega_3$$

$$p_{e_3} = \begin{bmatrix} L_1c_1 + L_2c_{12} + L_3c_{123} \\ L_1s_1 + L_2s_{12} + L_3s_{123} \\ 0 \end{bmatrix}$$

$$v_{e_3} = \begin{bmatrix} -L_1\dot\theta_1 s_1 - L_2(\dot\theta_1+\dot\theta_2)s_{12} - L_3(\dot\theta_1+\dot\theta_2+\dot\theta_3)s_{123} \\ L_1\dot\theta_1 c_1 + L_2(\dot\theta_1+\dot\theta_2)c_{12} + L_3(\dot\theta_1+\dot\theta_2+\dot\theta_3)c_{123} \\ 0 \end{bmatrix}$$

$$\omega_3 = \begin{bmatrix} 0 \\ 0 \\ \dot\theta_1+\dot\theta_2+\dot\theta_3 \end{bmatrix} \qquad I_3 = \begin{bmatrix} I_{xx_3} & 0 & 0 \\ 0 & I_{yy_3} & 0 \\ 0 & 0 & I_{zz_3} \end{bmatrix}$$

$\square$ $\theta_1 - \theta_{123} = -\theta_{23}$

$$\langle v_{c_3}, v_{c_3}\rangle = L_1^2\dot\theta_1^2 s_1^2 + L_2^2(\dot\theta_1+\dot\theta_2)^2 s_{12}^2 + L_3^2(\dot\theta_1+\dot\theta_2+\dot\theta_3)^2 \cdot s_{123}^2$$
$$+2\left(L_1L_2\dot\theta_1(\dot\theta_1+\dot\theta_2)s_1 s_{12} + L_1L_3\dot\theta_1\dot\theta_{123}\cdot s_1\cdot s_{123}\right.$$
$$\left. + L_2L_3\dot\theta_{12}\dot\theta_{123} s_{12}s_{123}\right)$$
$$+L_1^2\dot\theta_1^2 c_1^2 + L_2^2\dot\theta_{12}^2 c_{12}^2 + L_3^2\dot\theta_{123}^2 c_{123}^2$$
$$+2\left(L_1L_2\dot\theta_1\dot\theta_{12}c_1c_{12} + L_1L_3\dot\theta_1\dot\theta_{123}c_1c_{123}\right.$$
$$\left. + L_2L_3\dot\theta_{12}\dot\theta_{123}c_{12}c_{123}\right)$$

$$= L_1^2\dot\theta_1^2 + L_2^2\dot\theta_{12}^2 + L_3^2\dot\theta_{123}^2$$
$$+2\left[L_1L_2\dot\theta_1\dot\theta_{12}(\underbrace{s_1s_{12}+c_1c_{12}}_{\to c_2})\right.$$
$$+L_1L_3\dot\theta_1\dot\theta_{123}(\underbrace{s_1\cdot s_{123}+c_1c_{123}}_{\to c_{23}})$$
$$\left.+L_2L_3\dot\theta_{12}\dot\theta_{123}(\underbrace{s_{12}s_{123}+c_{12}c_{123}}_{\to c_3})\right]$$

$$\langle v_{c_3}, v_{c_3}\rangle = L_1^2\dot\theta_1^2 + L_2^2\dot\theta_{12}^2 + L_3^2\dot\theta_{123}^2 + 2\left(L_1L_2\dot\theta_1\dot\theta_{12}c_2 + L_1L_3\dot\theta_1\dot\theta_{123}c_{23}\right.$$
$$\left.+L_2L_3\dot\theta_{12}\dot\theta_{123}c_3\right)$$

$\boxed{KE_3}$

$$\boxed{KE_3 = \frac{1}{2}m_3\left[L_1^2\dot\theta_1^2 + L_2^2\dot\theta_{12}^2 + L_3^2\dot\theta_{123}^2 + 2\left(L_1L_2\dot\theta_1\dot\theta_{12}c_2 + L_1L_3\dot\theta_1\dot\theta_{123}c_{23} + L_2L_3\dot\theta_{12}\dot\theta_{123}c_3\right)\right] + \underline{I}_{zz3}\dot\theta_{123}^2}$$

<u>Where</u>, $\theta_{i..j} = \theta_i + \theta_{i+1} + \ldots + \theta_j$ and $\dot\theta_{i..j} = \dot\theta_i + \dot\theta_{i+1} + \ldots + \dot\theta_j$

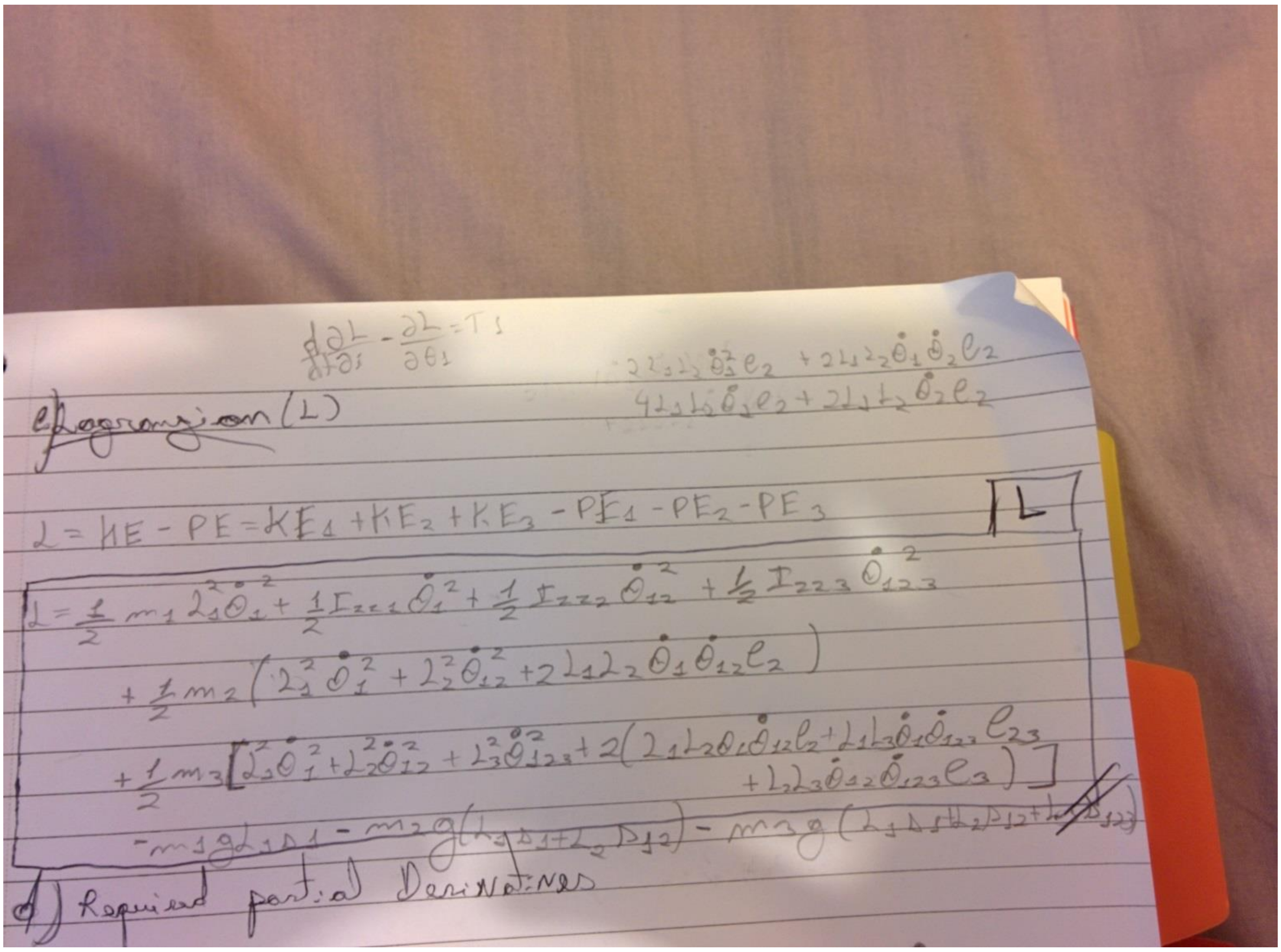
2) <u>Potential</u> <u>Energy</u>

$$PE = PE_1 + PE_2 + PE_3$$

$$\boxed{PE_1 = m_1 g L_1 s_1}$$

$$\boxed{PE_2 = m_2 g(L_1 s_1 + L_2 s_{12})}$$

$$\boxed{PE_3 = m_3 g(L_1 s_1 + L_2 s_{12} + L_3 s_{123})}$$



$\frac{d}{dt}\frac{\partial L}{\partial \dot\theta_1} - \frac{\partial L}{\partial \theta_1} = \tau_1$

$2 l_1 l_2 \dot\theta_1^2 c_2 + 2 l_1 l_2 \dot\theta_1 \dot\theta_2 c_2$

$4 l_1 l_2 \dot\theta_1 c_2 + 2 l_1 l_2 \dot\theta_2 c_2$

e) Lagrangian (L)

$L = KE - PE = KE_1 + KE_2 + KE_3 - PE_1 - PE_2 - PE_3$

$\boxed{L}$

$$\boxed{\begin{aligned} L = {} & \frac{1}{2} m_1 l_1^2 \dot\theta_1^2 + \frac{1}{2} I_{zz1} \dot\theta_1^2 + \frac{1}{2} I_{zz2} \dot\theta_{12}^2 + \frac{1}{2} I_{zz3} \dot\theta_{123}^2 \\ & + \frac{1}{2} m_2 \left(l_1^2 \dot\theta_1^2 + l_2^2 \dot\theta_{12}^2 + 2 l_1 l_2 \dot\theta_1 \dot\theta_{12} c_2 \right) \\ & + \frac{1}{2} m_3 \left[l_1^2 \dot\theta_1^2 + l_2^2 \dot\theta_{12}^2 + l_3^2 \dot\theta_{123}^2 + 2 \left(l_1 l_2 \dot\theta_1 \dot\theta_{12} c_2 + l_1 l_3 \dot\theta_1 \dot\theta_{123} c_{23} \right.\right. \\ & \left.\left. + l_2 l_3 \dot\theta_{12} \dot\theta_{123} c_3 \right) \right] \\ & - m_1 g l_1 s_1 - m_2 g (l_1 s_1 + l_2 s_{12}) - m_3 g (l_1 s_1 + l_2 s_{12} + \cancel{l_3 s_{123}}) \end{aligned}}$$

d) Required partial Derivatives

III. CONCLUSION

This report described how the solutions for question 1, 2, 3 and 4 were carried out. In question 1, the problems of forward kinematics and Jacobian computation were addressed. Question 2 proposed the inverse kinematics problem for planar robots. To handle the problem of singular configurations, the DLS inverse was used. Then, some letters were drawn using a 20-DOF robot arm. Still in question 2, a quick experiment was performed to show that the robot tries as best as it can to achieve its secondary objective q_{des}, even when it does not have enough DOF to do so. Question 3 proposed the derivation of the equation of motion for a 3-DOF planar robot using the Lagrangian formulation, this question had its letters a), b), c) and d) solved. Letter d) was solved by symbolic computation using Mathematica to compute the required partial derivatives from the Lagragian L, which was derived by hand. Also, an alternative symbolic derivation was computed using Mathematica, without considering any handmade derivation. Both derivations are in the appendices folder, whereas the handmade solutions for letters a), b) and c) can be found in the subsection E of this report. Both solutions were checked according the Siciliano's book and Lecture notes, and seemed to be correct. Finally, in question 4 two problems were addressed at the same time: inverse and forward dynamics. The solution of this problem for a planar robot enabled the simulation of 1 link and 3 link robots, whose results were correspondingly showed in this report.