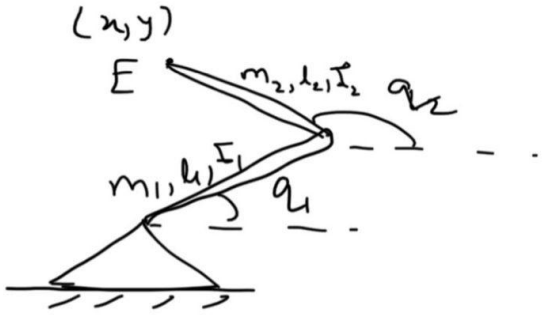
ITR
MINI PROJECT

Anavart Pandya
20110016

Task 0



From simple maths,

$$x = l_1 \cos q_1 + l_2 \cos q_2$$

$$y = l_1 \sin q_1 + l_2 \sin q_2 \quad \text{---(1)}$$

Further differentiating both sides,

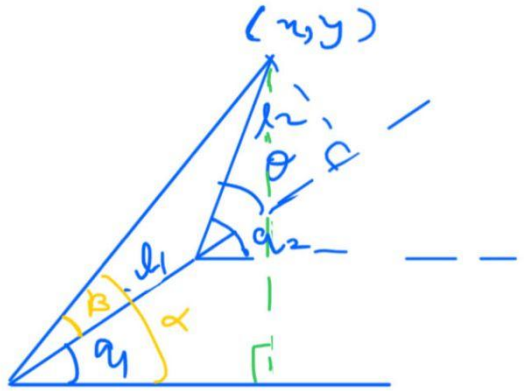
$$\dot{x} = -l_1 \dot{q}_1 \sin q_1 - l_2 \dot{q}_2 \sin q_2$$

$$\dot{y} = l_1 \dot{q}_1 \cos q_1 + l_2 \dot{q}_2 \cos q_2$$

$$\therefore \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -l_1 \sin q_1 & -l_2 \sin q_2 \\ l_1 \cos q_1 & l_2 \cos q_2 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \quad \text{---(2)}$$

solving for inverse kinematics,

given



$$\theta = \cos^{-1}\left(\frac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2}\right)$$

also, $q_1 = \alpha - \beta$

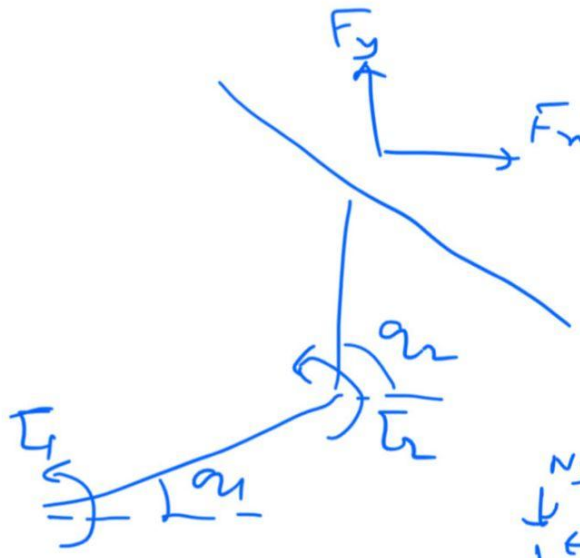
$$\tan\alpha = \frac{y}{x}$$

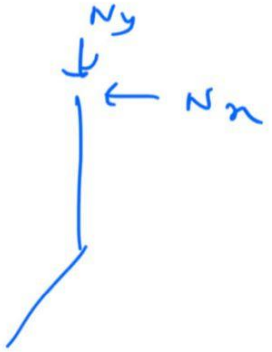
$$\tan\beta = \frac{l_2\sin\theta}{l_1+l_2\cos\theta}$$

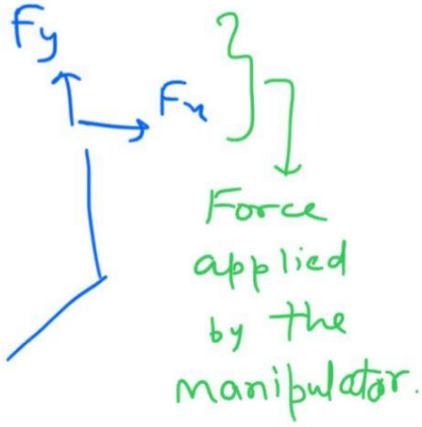
$$\therefore q_1 = \tan^{-1}\left(\frac{y}{x}\right) - \tan^{-1}\left(\frac{l_2\sin\theta}{l_1+l_2\sin\theta}\right)$$

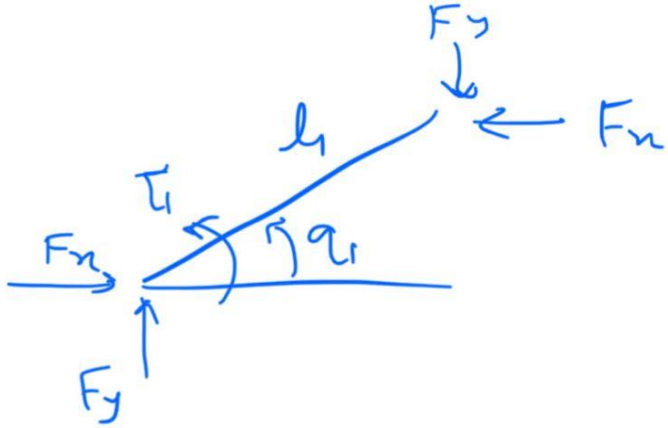
also, $q_2 = q_1 + \theta$

Torque - Force relation









ignoring gravity and balancing torque,

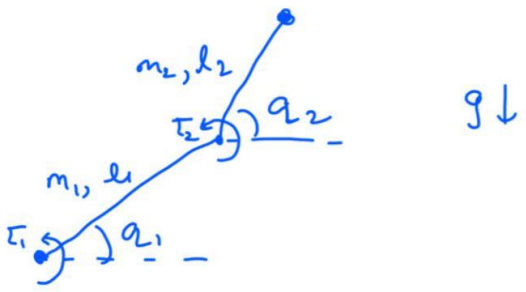
$$\tau_1 = -F_x l_1 \sin q_1 + F_y l_1 \cos q_1$$

$$\tau_2 = -F_x l_2 \sin q_2 + F_y l_2 \cos q_2$$

$$\therefore \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} -l_1 \sin q_1 & l_1 \cos q_1 \\ -l_2 \sin q_2 & l_2 \cos q_2 \end{bmatrix} \begin{bmatrix} F_x \\ F_y \end{bmatrix}$$

(4)

2R - manipulator Torque Relation Proof



$$L = K - V$$

$$K = \frac{1}{2}\left(\frac{1}{3} m_1 l_1^2\right)(\dot{q}_1)^2 + \frac{1}{2}\left(\frac{1}{12} m_2 l_2^2\right)(\dot{q}_2)^2 + \frac{1}{2} m_2 v_{c_2}^2$$

$$v_{c_2}^2 = (\dot{q}_1 l_1)^2 + \left(\dot{q}_2 \frac{l_2}{2}\right)^2 + 2\dot{q}_1 \dot{q}_2 l_1 \frac{l_2}{2} \cos(q_2 - q_1)$$

$$v_{c_2}^2 = (l_1 \dot{q}_1)^2 + \left(\frac{l_2}{2}\dot{q}_2\right)^2 + \dot{q}_1 \dot{q}_2 l_1 l_2 \cos(q_2 - q_1)$$

$$\therefore K = \frac{1}{2}\left(\frac{1}{3} m_1 l_1^2\right)(\dot{q}_1)^2 + \frac{1}{2}\left(\frac{1}{12} m_2 l_2^2\right)(\dot{q}_2)^2 + \frac{1}{2} m_2 \left((l_1 \dot{q}_1)^2 + \left(\frac{l_2}{2}\dot{q}_2\right)^2 + \dot{q}_1 \dot{q}_2 l_1 l_2 \cos(q_2 - q_1)\right)$$

$$V = m_1 g \frac{l_1}{2} \sin q_1 + m_2 g \left(l_1 \sin q_1 + \frac{l_2}{2} \sin q_2\right)$$

$$\therefore L = \frac{1}{2}\left(\frac{1}{3}m_1 l_1^2\right)(\dot{q}_1)^2 + \frac{1}{2}\left(\frac{1}{12}m_2 l_2^2\right)(\dot{q}_2)^2$$
$$+ \frac{1}{2}m_2\left(l_1\dot{q}_1)^2 + (\frac{l_2}{2}\dot{q}_2\right)^2 + \dot{q}_1\dot{q}_2 l_1 l_2 \cos(q_2 - q_1)\Big)$$
$$- m_1 g \frac{l_1}{2}\sin q_1 - m_2 g\left(l_1 \sin q_1 + \frac{l_2}{2}\sin q_2\right)$$

As we know, $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = T_i$

Calculating each term :

1. $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_1}\right)$:-

$$\frac{\partial L}{\partial \dot{q}_1} = \frac{1}{3}m_1 l_1^2 \dot{q}_1 + m_2 l_1^2 \dot{q}_1 + \frac{m_2}{2}\dot{q}_2 l_1 l_2 \cos(q_2 - q_1)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_1}\right) = \frac{1}{3}m_1 l_1^2 \ddot{q}_1 + m_2 l_1^2 \ddot{q}_1$$
$$+ \frac{m_2}{2}\ddot{q}_2 l_1 l_2 \cos(q_2 - q_1)$$
$$- \frac{m_2}{2}\dot{q}_2 (\dot{q}_2 - \dot{q}_1) l_1 l_2 \sin(q_2 - q_1)$$

2. $\frac{\partial L}{\partial q_1}$:—

$$\frac{\partial L}{\partial q_1} = \frac{1}{2} m_2 \dot{q}_1 \dot{q}_2 l_1 l_2 \sin(q_2 - q_1) - m_1 g \frac{l_1}{2} \cos q_1 - m_2 g l_1 \cos q_1$$

3. $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_2}\right)$:—

$$\frac{\partial L}{\partial \dot{q}_2} = \frac{1}{12} m_2 l_2^2 \dot{q}_2 + \frac{1}{4} m_2 l_2^2 \dot{q}_2 + \frac{m_2}{2} \dot{q}_1 l_1 l_2 \cos(q_2 - q_1)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_2}\right) = \frac{1}{12} m_2 l_2^2 \ddot{q}_2 + \frac{1}{4} m_2 l_2^2 \ddot{q}_2 + \frac{m_2}{2} \ddot{q}_1 l_1 l_2 \cos(q_2 - q_1) - \frac{m_2}{2} \dot{q}_1 (\dot{q}_2 - \dot{q}_1) l_1 l_2 \sin(q_2 - q_1)$$

4. $\frac{\partial L}{\partial q_2}$:—

$$\frac{\partial L}{\partial q_2} = -\frac{1}{2} m_2 \dot{q}_1 \dot{q}_2 l_1 l_2 \sin(q_2 - q_1) - m_2 g \frac{l_2}{2} \cos q_2$$

$$\Rightarrow \tau_1 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_1}\right) - \frac{\partial L}{\partial q_1}$$

$$\tau_1 = \frac{1}{3} m_1 l_1^2 \ddot{q}_1 + m_2 l_1^2 \ddot{q}_1 + \frac{m_2}{2} \ddot{q}_2 l_1 l_2 \cos(q_2 - q_1) - \frac{m_2}{2} \dot{q}_2 (\dot{q}_2 - \dot{q}_1) l_1 l_2 \sin(q_2 - q_1) - \frac{1}{2} m_2 \dot{q}_1 \dot{q}_2 l_1 l_2 \sin(q_2 - q_1) + m_1 g \frac{l_1}{2} \cos q_1 + m_2 g l_1 \cos q_1$$

$$\tau_2 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_2}\right) - \frac{\partial L}{\partial q_2}$$

$$\tau_2 = \frac{1}{12} m_2 l_2^2 \ddot{q}_2 + \frac{1}{4} m_2 l_2^2 \ddot{q}_2$$

$$+ \frac{m_2}{2} \ddot{q}_1 l_1 l_2 \cos(q_2 - q_1)$$

$$- \frac{m_2}{2} \dot{q}_1 (\dot{q}_2 - \dot{q}_1) l_1 l_2 \sin(q_2 - q_1)$$

$$+ \frac{1}{2} m_2 \dot{q}_1 \dot{q}_2 l_1 l_2 \sin(q_2 - q_1)$$

$$+ m_2 g \frac{l_2}{2} \cos q_2$$

Note: The simulations of all the tasks are done in pygame.

Task 1

In Task 1, the are two files. One is without dynamics named as **Task1_withoutdynamics.py** and the other file is with dynamics named as **Task1_withdynamics.py**. The manipulator in both the files is trained to follow the mouse cursor. When you run the code, a pygame window will open. You will observe a small circle around the cursor. Take the cursor to the desired position. The manipulator in both files will go to the location of the cursor. In case, the cursor position is out of the workspace of the manipulator, the manipulator will reach the closest point (to the cursor location) that is in the workspace of the manipulator.

The speed of the manipulator implemented without dynamics can be changed by changing the value of variable **'N'**. Higher the value of **N**, lower will be the speed and vice-versa. For the changing the speed of the manipulator implemented with dynamics, the value of variable '**tf'** needs to be changed. Higher the value of 'tf' higher will be the speed and vice-versa. Also here, variable **'fps'** plays a major role. Higher the value of **fps** smoother will the motion and vice-versa. If the **fps** is too high and exceeds the maximum permissible frame rate per second of your device, the speed of the manipulator will get reduced.

The simulation of with and without dynamics may not seem visually different at the first insight. But if observed carefully, the motion of the manipulator is much smoother than that of without dynamics. This is because the velocity smoothly reduces in the case with dynamics whereas in case of without dynamics, the velocity abruptly becomes zero after reaching the target location. Another major difference in both the approaches is that the motion without dynamics is 100% accurate whereas the motion with dynamics has slight error when calculating the angles. This is due to the fact that in the case with dynamics, the angles are calculated by solving ordinary differential equations, which will account for some error, even if the time interval taken is much smaller.

There was no change except for the speed in the case of manipulator without dynamics. The high speed and low speed motions seemed almost similar. But it was not with the case of manipulator with dynamics. The manipulator behaved very differently at very high speed and very low speed. At low speed, the manipulator motion was very smooth and was almost resembling a real manipulator. But high speeds, the manipulator motions seemed rough, abrupt and unrealistic. This was due to the fact that the no.of points between the final and initial point got reduced and therefore, the results of ode were not very accurate.

Also in case of manipulator with dynamics, the concept of **trajectory interpolation** was used.

Task 2

For this task, run **Task2.py** file. In this task, the magnitude of force, wall location, wall orientation and the point of application of force on the wall has to be decided by the user. First click on mouse will stop the manipulator (if moving) and allow you to move the cursor freely in the space. After the first click, take your cursor to the your desired position where you want your wall's first end point to be. Once you reach that point, click the mouse for the second time. This will the fix the location of first end point of the wall. After, second click you will observe that you can again move your cursor freely but this time a line will be shown between your cursor and the first end point of the wall previously fixed by you. Move the cursor to the location where you want place the second end point of the wall and then click the mouse button for the third time. After the third click, you now have to move the cursor to the desired position on the wall where you want the force to act and click the mouse for the fourth time.

The manipulator will go to the target location and will apply the force of the given magnitude in the direction perpendicular to the wall. As the manipulator reaches the wall, and applies the given force, the torques applied by the motors for applying the force are shown on the window. The forces in x and y direction of the ground reference frame are also shown on the window. Also, the direction of the force applied is shown by a vector (represented by a green line and vector head as a red circle) which start from the target location. In this task, all motions of the manipulator are happening by taking dynamics into account. The speed of the manipulator can be changed by repeating the same steps as mentioned in **with dynamics manipulator in task1**. The desired magnitude of force can be set by setting the value of variable **'F'**.

Task 3

In Task 3, there are two files named **Task3_location_method.py** and **Task3_drag_method.py.** The only difference in the two files is the method of giving the input target location to the manipulator. In the first location method file, the desired location is given to the manipulator by a mouse click at the desired location. The manipulator moves to that location with taking dynamics into consideration (including the spring effects) just as in task 1. When the manipulator reaches to the target location, it recoils back to the equilibrium position and the motion seems as if a virtual spring is attached to the end effector.

In the drag method file, the user is free to drag the manipulator from the initial equilibrium position to any arbitrary position and release from that position. When the manipulator will be released, it will reach the equilibrium position and the motion will seem as if a virtual spring is attached to the end effector. Here also, all the motion of the manipulator take dynamics into account. Note that, to drag the manipulator, click the mouse one time (donot click and hold the mouse button, just click it). After clicking, you will observe that the location of end effector becomes the same as your current cursor location. Once you reach your desired location, click the mouse again. You will observe that the end effector reaches back to the equilibrium position.

As there were very abrupt results and an infinite motion in considering only natural spring effects, damping is also introduced here. Due to the damping, the motion of the manipulator becomes very smooth especially near the equilibrium point. The values of damping coefficient (denoted by variable '**b_'** in both code files) and the spring constant (denoted by variable '**k'** in both code files) can be altered to observe different kinds of motion. Some of the suggested values for (b_,k) for smooth motion of the manipulator are (3,100), (15,1000) and (30, 1000). Note that, you may encounter very abrupt motions of manipulator if the ratio of b_ is to k is not appropriate. The location of the equilibrium point can be changed by providing different values to **'x_neutral'** and **'y_neutral'** variables in both the files.

Task 4

Run the **Task4.py** file to run the task. To calculate the workspace we need to know all points through which the end effector can pass through. Theoretically, there infinite no.of points, but numerically, it is upto us that how much close we want the points on the plot to be.

The points (x and y coordinates) are calculated from the corresponding angles q1 and q2. I.e., many values of q1 and q2 are considered with dq1 and dq2 difference between two consecutive angles and corresponding x and y coordinates are calculated and plotted on the graph. q1 and q2 are increased from 35 to 145 degrees gradually and all angles between these two values that are at a difference of dq1 and dq2 are taken into account.

Note that, the user can change the values of dq1 and dq2. The user can also change other parameters such as length and location of the ground point. Lower the values of dq1 and dq2, more no.of points can be plotted on the graph.