MEGN 544 ROBOT MECH: KINEMATICS, DYNAMICS AND CONTROL

PROJECT REPORT

Mohammed Aun Siddiqui
10834112

OPERATIONAL SPACE INVERSE DYNAMICS CONTROL

1.Introduction

Robots are autonomous or semi-autonomous machines that are capable of carrying out a series complex actions automatically. These are mostly programmable by a computer and are guided by an external control device or control may be embedded within it. To enable a robot to do various movements a lot of work goes in designing it, taking into account its motion kinematics, dynamics, and control.

In this project we are trying to control a 3 linked robotic arm. The problem given to us is to obtain efficient control performance on a 3 linked arm, which should be able to move in the given Cartesian trajectory. This project was divided into 3 parts:

- In the first part, we modelled the robot kinematics using Matlab.
- The second part dealt with robot dynamics.
- The third part deals with using the Matlab functions (that were coded in the first two parts) to design a controller for the given arm using Simulink.

For controlling the robotic arm, we implemented Operational Space Inverse Dynamics Control, and to remove any discrepancy and unknown mismatches Sliding Mode Control was also augmented to give a robust controller. The overall performance is dependent on the gains. By increasing the gains, we increase the system performance but after a certain value the system starts chattering. This chatter can be overcome by implementing sliding mode control making the system all the more robust.

2.Model Description

The arm given to us along with its DH parameters as well as the dimensions of the links. The arm is assumed to be made of aluminum that has a density of 2700 kg/m^3. Using the formula that density= mass/ volume we calculated the mass of each link. As well as the center of mass w.r.t. the next link.

LINK NO	LENGTH(M)	A	ALPHA	D	THETA	MASS(KG)	RCOM(M)
1	0.25	0	pi/2	0.25	*	5.30	[0;-0.125;0]
2	1	1	0	0	*	21.20	[-0.5;0;0]
3	0.5	0.5	0	0	*	10.60	[-0.25;0;0]

Where * indicates the control parameter.

After this we assumed the links to a be perfect cylinders and calculated the inertia tensors, which turned out to be as:

For Link 1:

$$\begin{bmatrix} 0.0309 & 0 & 0 \\ 0 & 0.00662 & 0 \\ 0 & 0 & 0.0309 \end{bmatrix}$$

For Link 2:

$$\begin{bmatrix} 0.0265 & 0 & 0 \\ 0 & 1.7799 & 0 \\ 0 & 0 & 1.7799 \end{bmatrix}$$

For Link 3:

$$\begin{bmatrix} 0.01325 & 0 & 0 \\ 0 & 0.2275 & 0 \\ 0 & 0 & 0.2275 \end{bmatrix}$$

After doing that I fed all the values to the createLink function that creates a structure of the link parameters. Since all the links are rotary the isRotary was true for all links.

The other functions that were used for implementing the arm were:

- DHtransform: Returns the homogenous transforms for the given DH parameters
- dhfwdKine: getting the forward kinematics i.e. positions from joint space to operational space
- velocityJacobian: for getting the velocity Jacobian matrix
- newtonEuler: For the computing the joint torques

3.Controller Description

As the names suggest we implement the controller in operational space that is the 3D space and not in joint space that involves θs. The trajectory generator block provides us with the desired velocity, acceleration and position of the end-effector. We then proceed to implement OSIDC and try to match the desired trajectory to the current trajectory by changing the joint-torque which is calculated by newtonEuler. For calculating error between desired and actual, from the plant we get values for θ and θ' to convert this back into operational space we use dhfwdKine and velocityJacobian. The control law is for this is given as:

$$\boldsymbol{y} = \boldsymbol{J}_A^{-1}(\boldsymbol{q})\left(\ddot{\boldsymbol{x}}_d + \boldsymbol{K}_D\dot{\tilde{\boldsymbol{x}}} + \boldsymbol{K}_P\tilde{\boldsymbol{x}} - \dot{\boldsymbol{J}}_A(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\right)$$

Neglecting terms for gravity, centripetal and Coriolis, and friction. The highlighted portion in the block diagram is the sliding mode control.

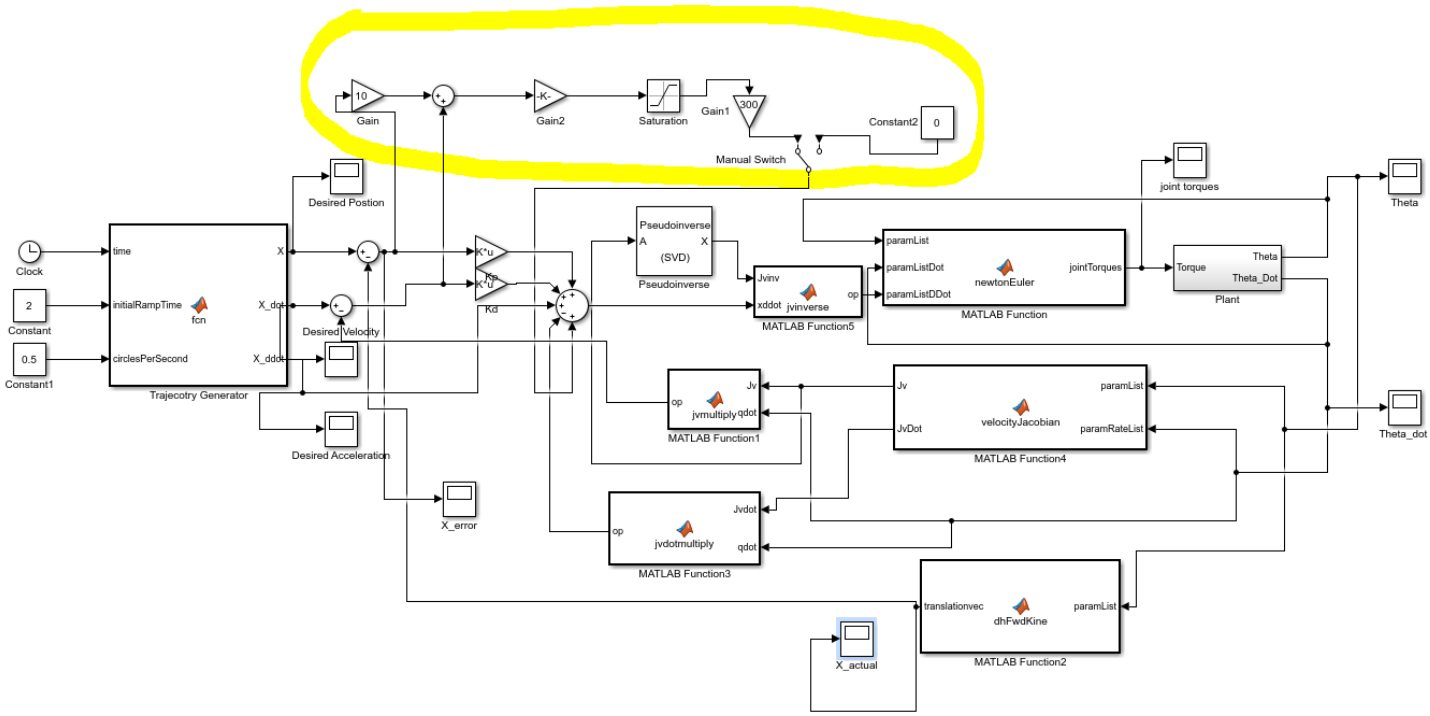

fig 3.1: Block Diagram for operational space inverse dynamics controls

The gains Kp and Kd are chosen by the taking values of settling time as 4 and percentage overshoot as 10 then plugging them into:

$$\zeta = \frac{-\ln(\%OS/100)}{\sqrt{\pi^2 + \ln^2(\%OS/100)}} \qquad T_s = \frac{4}{\zeta\omega_n}$$

We can then calculate $K_d = 2\zeta\omega_n$, $K_p = \omega_n^2$

Kp is the proportional gain and kd is the derivative gain. Since the natural frequency is not known for the system we chose the gain by trial and error based on the system performance. Sliding mode is useful for removing chatter and makes the system robust for higher order nonlinear systems.

4.ControllerPerformance

The performance of the controller (without sliding mode) with respect to trajectory tracking and RMS error of the position has been discussed in the next section. To summarize it, higher gains give lesser error and better tracking. In the next section we realize that for gains Kp=500 and Kd=800 the system has a chatter problem (see fig 5.2), if for some reason we cannot increase the gains, to reduce chatter we augment sliding mode control to the system. The performance for sliding mode control has been discussed by keeping gains constant" a" constant at 10 and Fmax constant at 300, and we change epsilon. We then plot for RMS error and Joint torques.

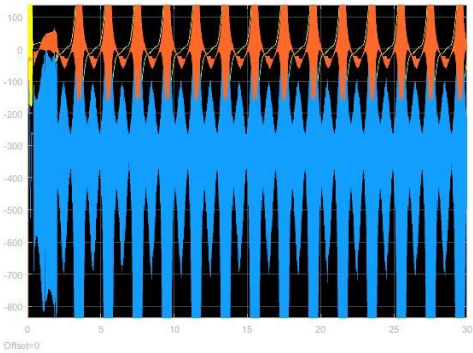

Fig 4.1: Joint Torque for epsilon= 0.1

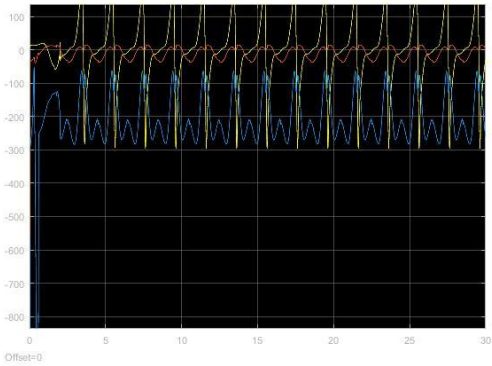

Fig 4.2: Joint Torque for epsilon= 10

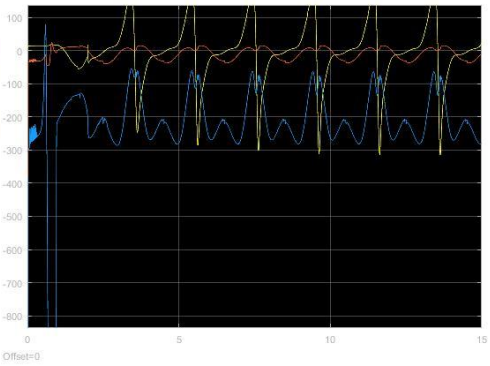

Fig 4.3: Joint Torque for epsilon= 100

We see that by reducing epsilon by decreasing there is a lot of chatter present but the RMS error is the lowest. This system will be best suited to fast systems that do not require a lot of computation time. As we increase epsilon the chatter decreases but the RMS error increases. This is clearly demonstrating a trade-off between chatter and RMS error.

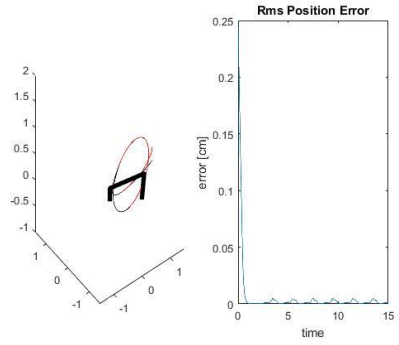

Fig 4.4: Joint Torque for epsilon= 0.1

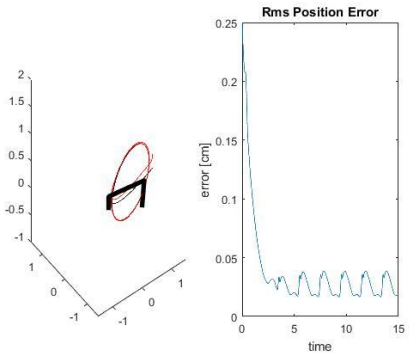

Fig 4.5: Joint Torque for epsilon= 10

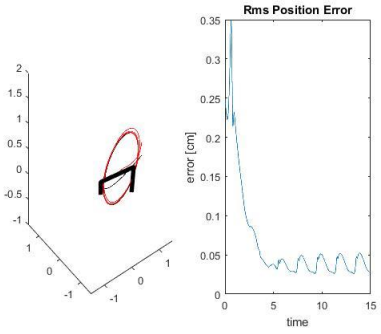

Fig 4.6: Joint Torque for epsilon= 100

5. Discussion

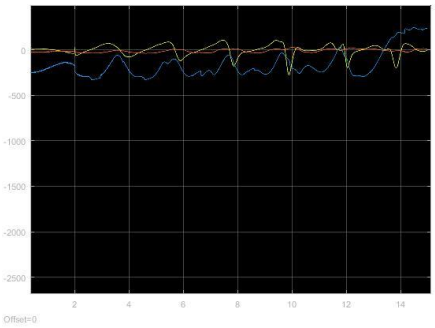

Fig 5.1: Joint Torque for Kp=40 and Kd=75

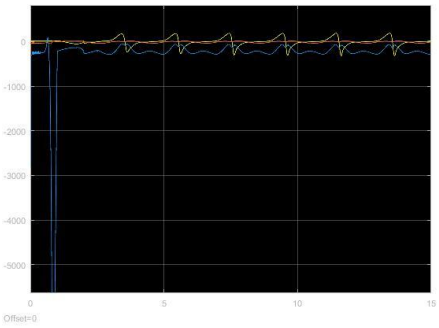

Fig 5.2: Joint Torque for Kp=500 and Kd=800

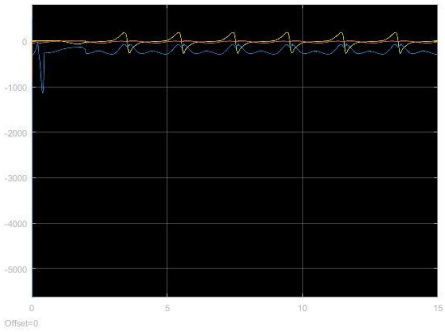

Fig 5.3: Joint Torque for Kp=1150 and Kd= 1350

From the graphs above we compare systems without implementing sliding mode, from the plots we can see that as we increase the gain the system gets sped up and reaches steady state faster. The fastest trajectory that the controller is in fig 5.3. There is no overshoot in the middle as in the case shown in fig5.2 and the RMS error is reasonably low. Another thing that can be noticed is that there is no chatter present for fig 5.3. The RMS error decreases after increasing Kp and Kd. Hence, the overall performance is increased by increasing the gains up to a certain extent. After that there is a lot of chatter that is present in the system that can't be controlled even with sliding mode.

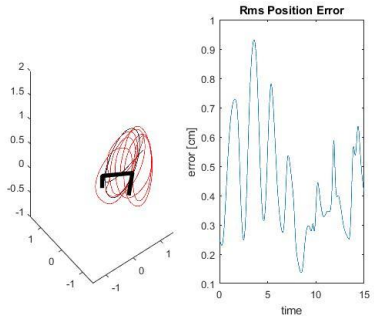

Fig 5.4: RMS error for Kp=40 and Kd=75

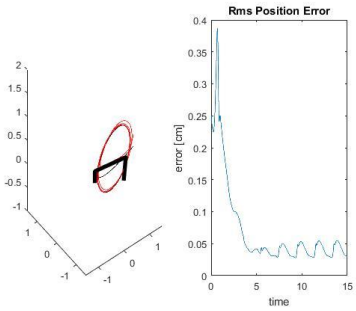

Fig 5.5: RMS error for Kp=500 and Kd=800

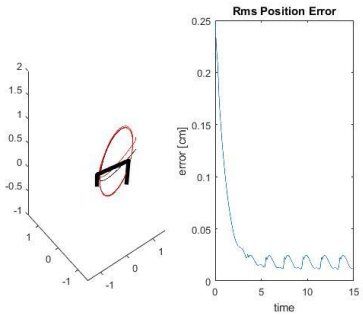

Fig 5.6: RMS error for Kp=1150 and Kd= 1350

If we saturate the joint torque, that is give it some upper and lower bound the system takes a lot of time to come down to its steady state and gives a lot RMS position error. This can be seen clearly from fig 5.1 and fig 5.4. The system does require an overshooting high torque at the start to stabilize itself and reach steady state that can be seen from the remaining figures. For tracking the desired trajectory, the best performance is got by increasing the gains. Trajectories for really high gains that cannot be followed reasonably have been left out of this discussion. I found that the gains above Kp=1150 and Kd=1350 couldn't follow a reasonably decent trajectory.

Conclusion:

The performance of the controller increases by increasing the gains till a certain value. The RMS error and chatter in joint torque reduces as see from section 4. But once the threshold value for the gain is crossed, the performance starts to decrease as well. The system starts to give positional error as well as it starts chattering. To overcome this chatter, we need to implement sliding mode control that reduces both the RMS error as well as the chatter in the join torque. To make the controller more accurate we can account for these terms as well and them to our controller and make necessary changes in our newtonEuler. From this project we can conclude that for increasing the performance the gains have to be increased till a certain value. To further increase the system performance, we need to still increase the gain and augment sliding mode control which makes the system more robust, lesser prone to errors and faster. It also takes care of the noise that a system gets because of increasing the gains.

This project and control system work fine for a 3-linked arm but the most of the arms present nowadays are 6 linked> in the future I would like to extend my work for these arms as well and also implement real scenario by adding essential boundary conditions are missing such as centripetal, Coriolis and friction. This would make my system even more robust on a real arm.

Project Impression:

Wubbalubbadubdub (<---actually a word) could sum it up.

Well., I got to learn a lot from it.