SSY156 Modelling and control of mechatronic systems

Lab Assignment 3

Daniel Söderqvist (danisode)

March 17, 2023

Exercise 1

In this task we are suppose to construct the regressor matrix Y_r and finish the script "Model/Yr_abbIRB4.m". The first thing in order to finish the script is to try divide τ into Θ and Y_r which is made in the following manner described down below (1): We can see that by first calculating τ with all the given data we can divide τ into

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_\mathbf{r} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}_\mathbf{r} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau} = \mathbf{Y}_r\boldsymbol{\Theta}$$

$$\boldsymbol{\tau} = \mathbf{Y}_r\boldsymbol{\Theta} \longrightarrow \mathbf{A}\mathbf{x} = \mathbf{b}$$

$$\mathbf{Y}_r(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) = \begin{bmatrix} \ddot{q}_{1_r}, & \ddot{q}_{1_r}c_2 - \dot{q}_{1_r}\dot{q}_2c_2s_2 - \dot{q}_{2_r}\dot{q}_1c_2s_2, & 2\ddot{q}_{1_r}c_2 - \dot{q}_{1_r}\dot{q}_2s_2 - \dot{q}_1\dot{q}_{2_r}s_2, & c_1, & c_1c_2 \\ 0, & \ddot{q}_{2_r} + \dot{q}_{1_r}\dot{q}_1c_2s_2, & \dot{q}_{1_r}\dot{q}_1s_2, & 0, & s_1s_2 \end{bmatrix}$$
$$\boldsymbol{\Theta}(m_i, L_i, g) = \begin{bmatrix} L_1^2m_1^2 + L_1^2m_2^2 \\ L_2^2m_2^2 \\ L_1L_2m_2 \\ L_1g(m_1+m_2) \\ L_2gm_2 \end{bmatrix}$$

Figure 1: Equations that shows the approach to find Θ and Y_r, the equations are taken from the lecture notes.

one representation each for Θ and Y_r depending on which terms that it consists of. In Θ we want all terms depending on any lengths, masses, inertia's and viscous forces and the rest of the terms in the regressor matrix Y_r. By using the command coeffs() in matlab and then clear out all duplicates with the command unique() we received the Θ vector which had the length $\underline{p = 39}$. The other half of the output from coeffs() had the terms belonging to the regressor matrix but in order to organize the matrix so that the equality $\tau = Y_r\Theta$ is fulfilled we needed a nested loop to both find the right value and the corresponding position in the regressor matrix. The regressor matrix have the dimensions $\underline{Y_r = n \textbf{ x } p = 4 \textbf{ x } 39}$ due to the amount of joints and the length of the Θ vector. The function "ToRegressor.m" takes the matrices M and C and the G vector as input and gives the Θ vector and the regressor matrix Y_r consisting of both the ordinary and beta-values part. The output is printed as a symbolical vector and matrices to the command window when running the script "main.m" These matrices were copied to the function "Model/Yr_abbIRB4.m" that we were supposed to finish. The results will not be printed here but run "main.m" if you want to see the symbolical results. "See attached code "ToRegressor.m" for the approach of finding the results."

Exercise 2

In this task we need to find the coefficients in the following given equations:

$$\begin{gathered} P(i,t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \\ Pp(i,t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \\ Ppp(i,t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \end{gathered} \tag{1}$$

The coefficients are found solving the following equation:

$$Ax = b \tag{2}$$

Where the A matrix describes the amount of each a term, the b vector describes the known variables and the x vector describes the different terms we want to find out.

$$\underbrace{\begin{bmatrix} 1 & t_{ini} & t_{ini}^2 & t_{ini}^3 & t_{ini}^4 & t_{ini}^5 \\ 1 & t_{end} & t_{end}^2 & t_{end}^3 & t_{end}^4 & t_{end}^5 \\ 0 & 1 & 2t_{ini} & 3t_{ini}^2 & 4t_{ini}^3 & 5t_{ini}^4 \\ 0 & 1 & 2t_{end} & 3t_{end}^2 & 4t_{end}^3 & 5t_{end}^4 \\ 0 & 0 & 2 & 6t_{ini} & 12t_{ini}^2 & 20t_{ini}^3 \\ 0 & 0 & 2 & 6t_{end} & 12t_{end}^2 & 20t_{end}^3 \end{bmatrix},}_{A} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} P_{ini}(i) \\ P_{end}(i) \\ Pp_{ini}(i) \\ Pp_{end}(i) \\ Ppp_{ini}(i) \\ Ppp_{end}(i) \end{bmatrix}}_{b}, \tag{3}$$

When solving this we end up with the following equations:

$$\begin{gathered} P(i,1) = P_{ini}(i) + (P_{end}(i) - P_{ini}(i)) * (6 * r_5 - 15 * r_4 + 10 * r_3) \\ Pp(i,1) = (P_{end}(i) - P_{ini}(i)) * (30 * r_4 - 60 * r_3 + 30 * r_2)/Fac \\ Ppp(i,1) = (P_{end}(i) - P_{ini}(i)) * (120 * r_3 - 180 * r_2 + 60 * r)/Fac^2 \end{gathered} \tag{4}$$

Where:

$$\begin{gathered} D_t = t - t_{ini} \\ Fac = t_{end} - t_{ini} \\ r = D_t/Fac \\ r_2 = r * r \\ r_3 = r_2 * r \\ r_4 = r_3 * r \\ r_5 = r_4 * r \end{gathered} \tag{5}$$

The finished equations were put into the following function to finish it: "Model/Pol5th.m" which in turn finishes the asked script "trajGen_abbIRB4.m".

Exercise 3

In this task we are suppose to finish the function "Control/Qpr_abbIRB4.m". To finish this function everything needed is given in the assignment PM so by firstly defining the different control parameters K_d, K_i and K_p as decoupled parameters. The decoupled parameters can then be used in and to define the final asked equations:

$$\begin{aligned} \dot{q}_r &= \dot{q}_d - K_p \Delta q - K_i \int \Delta q \\ \ddot{q}_r &= \ddot{q}_d - K_p \Delta \dot{q} - K_i \Delta q \end{aligned} \tag{6}$$

All parameters and the final equation are put into the function which finishes it.

Exercise 4

In this task we are supposed to finish the functions "Control/Sq_abbIRB4.m", "Control/thetaHatDot_abbIRB4.m" and "Control/tauAdaptivePIDR.m" The first function is finished by defining joint velocity error which is simply the difference between $\dot{q}$ and $\dot{q}_r$:

$$S_q = \dot{q} - \dot{q}_r \tag{7}$$

The second function is finished by defining $\dot{\Theta}$ and to do this we use the given formula:

$$\dot{\Theta} = -\Gamma^{-1} Y r^{\top} S q \tag{8}$$

The third and last formula is finished by defining τ which is done with the following formula:

$$\tau = -K_d S_q + Y_r \dot{\Theta} \tag{9}$$

With all these equations we can finish up all of the 3 functions.

Exercise 5

In this task we are supposed to simulate the robot from a initial state q_0 to the desired position $q_d s$. Both the initial and desired position is set to be 0. Lastly we are supposed to tune the control parameters: K_p, K_i and K_d which is done by simulating multiple time while changing the parameters accordingly from simulation to simulation. We ended up after tuning with the following control parameters:

$$K_p = \begin{bmatrix} 45 & 60 & 60 & 75 \end{bmatrix}$$
$$K_i = \begin{bmatrix} 7 & 5 & 5 & 8 \end{bmatrix}$$
$$K_d = \begin{bmatrix} 1.5 & 1 & 1 & 2 \end{bmatrix}$$
$$\Gamma = 1$$

The result from using this control parameters is the following where the first 3 figures (2) (3) (4) are showing some clippings of different places in the progression throughout the simulation. These figures are followed by a figure showing the plot of S_q, Δq, q vs q_d, Θ and $\dot{\Theta}$ (5).

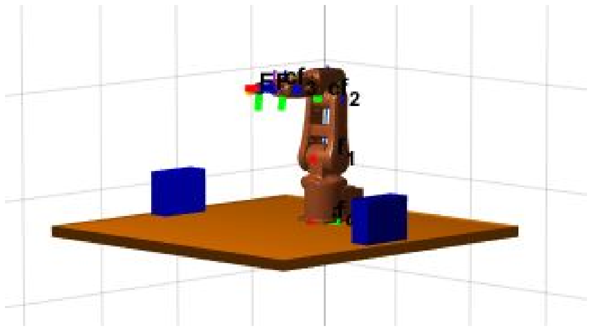

Figure 2: Plot of the robot almost the initial position.

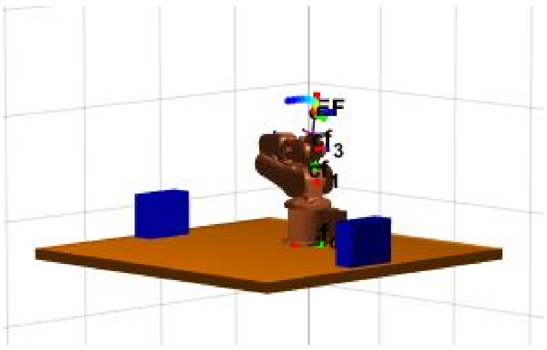

Figure 3: Plot of the robot in the middle of the simulation.

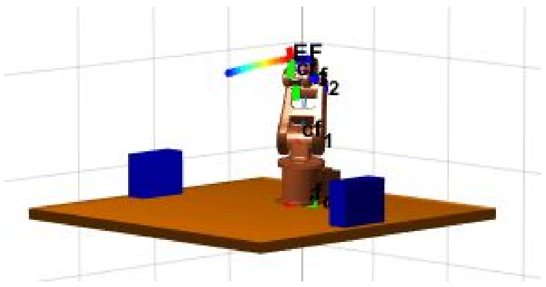

Figure 4: Plot of the robot in the middle of the simulation.

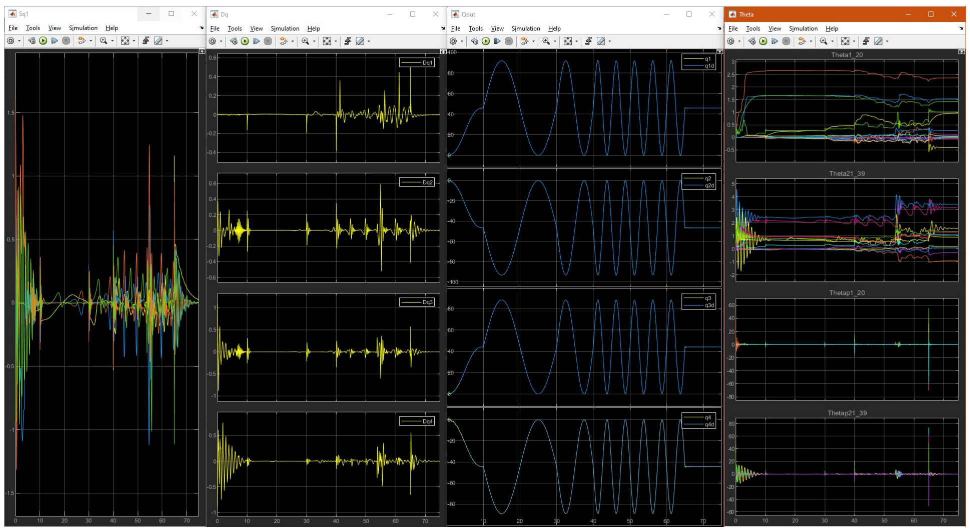

Figure 5: Plots showing the results of the simulation where the following is showed S_q, Δq, q vs q_d, Θ and $\dot{\Theta}$ from left to right respective.

From the results we can see with the actual control parameters we are able to put down the error where the controller with the feedback combined with the parameters actually is able to almost imitate the wanted result. We can see that in the beginning the curve is oscillating more to become almost stable after a few seconds into the simulation. And then start to fluctuate again when we are going from the different stages in the simulation in the transitions. The controller is after some rough transitions able to handle them all and stabilize the robot.

Exercise 6

In this task we are supposed to instead work with another robot that have 7 joints instead of 4. We are supposed to finish the function "KinematicControl-NullSpace/desired_op_trajectory.m" and to do this we were given many already predefined functions to help us just finish the last parts in order to simulate the robot. The exercise starts with a "Todo" to finish the following function: "Pol5th3DOF.m" but since it is the same equations as the similar function for the other robot we just copied the results from that one see exercise 2. When this function is done the task is to find the right HT, extract the rotational matrix and position vector convert to Euler angles, calculate the right trajectory and finally convert back to the rotational matrix from the Euler angles. To do the conversion and extracting so on the functions already predefined was used. Therefore with all of this done the function asked for in the beginning of this task was finished.

Exercise 7

In this task we are supposed to finish the function "Control/kinematicCtrlOpNS.m" and to this we are suppose to define equations for calculating the joint velocity with and without a null-space. As well as in the last task we got some functions to ease up the process. Using all these functions along with the given formulas in the assignment PM we started by extracting HT matrices, extracting, rotational and position data. Then define the velocity transformation which in turn leads to that we can define the analytical Jacobian. The analytical Jacobian along with the control parameter K_p the pose errors and the null space we could finally define the joint velocity using task error with and without a null-space:

$$q = J_{ef}^{0\#}(K_p \Delta x_{ef}^0) + N(K_{q_d}\Delta q + K_{q_d}\Delta\dot{q}) \tag{10}$$

Where the last term is the null-space which is considered when using the null-space and removed when not considering. With all this known we could finish the last function before simulating the new robot with a the new trajectory.

Exercise 8

In this task we are supposed to simulate the robot with trying different parameters like different primary task dimensions m, with and without a null-space,if we want smooth transitions and lastly if we want the trajectory to be a circle or a static point. We are supposed to pick a initial value where we have singularity free position which means a position where we don't loose 1 or more degrees of freedom. This is calculated by calculating the determinant of the jacobian matrix. If the value is 0 then we know we have singularity free position. We used in our case $det(J) < 0.01$ as reference to decide if we have a singularity free position or not. This inequality can be stricter or kinder depending on the case but we choose 0.01 as limit. Then we tested some initial values and found $Q = \begin{bmatrix} \frac{\pi}{8} & \frac{\pi}{8} & \frac{\pi}{8} & \frac{\pi}{8} & \frac{\pi}{8} & \frac{\pi}{8} & \frac{\pi}{8} \end{bmatrix}$ to be a singularity free position. (There is a test for this in the "main.m" where we can try different initial positions and different limits, see attached code: "main.m".) The first test was with all parameters like in the assignment PM $m = 3$ and all switches on. Gave the following results (6) (7) (8) (9) (10) (11):

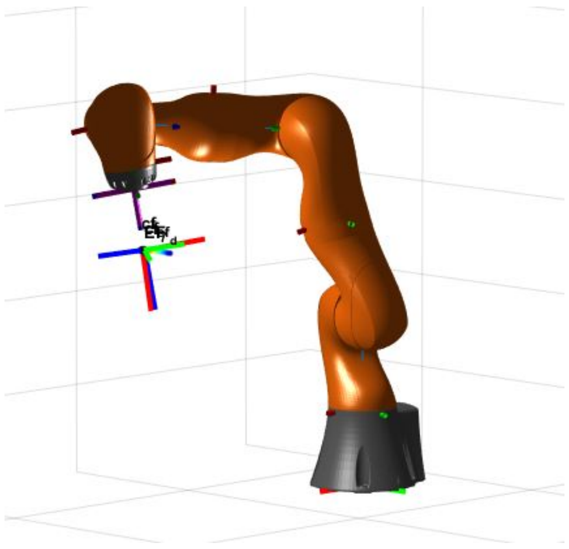

Figure 6: Plot of the robot in the middle of the simulation.

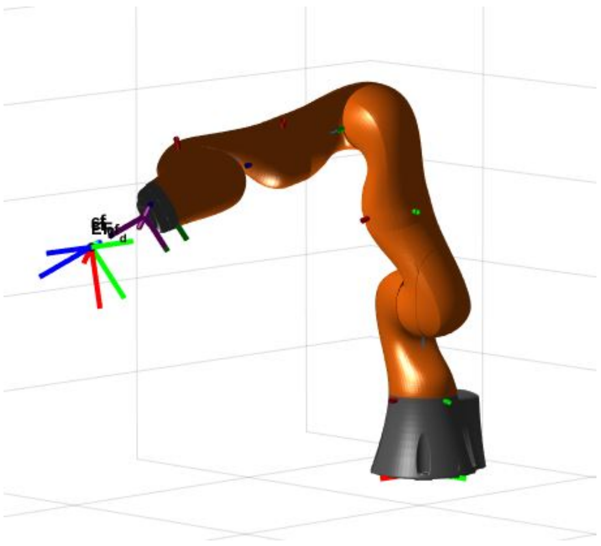

Figure 7: Plot of the robot in the middle of the simulation.

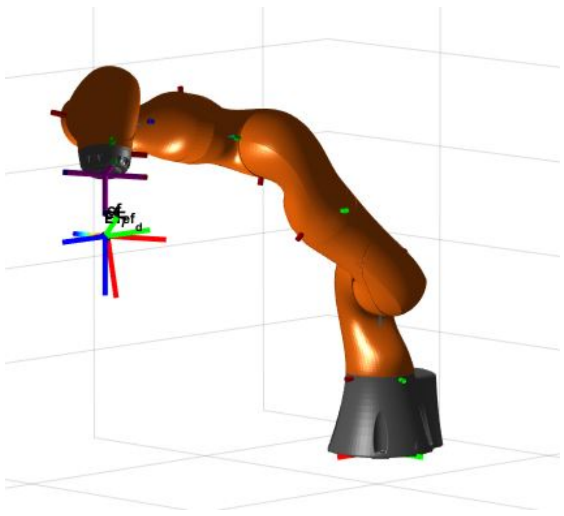

Figure 8: Plot of the robot in the middle of the simulation.

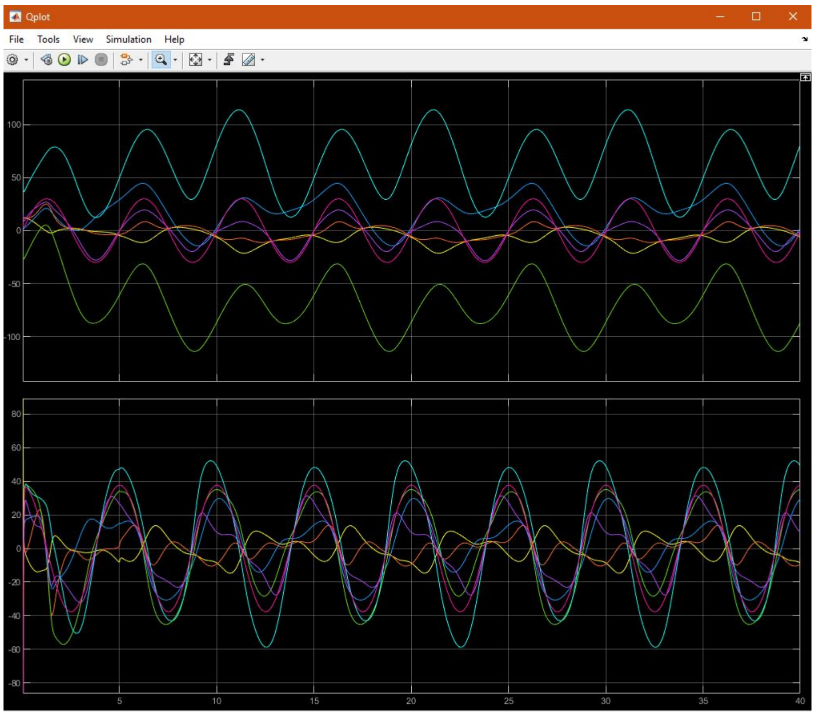

Figure 9: Plots showing the positions of the joints of the robot.

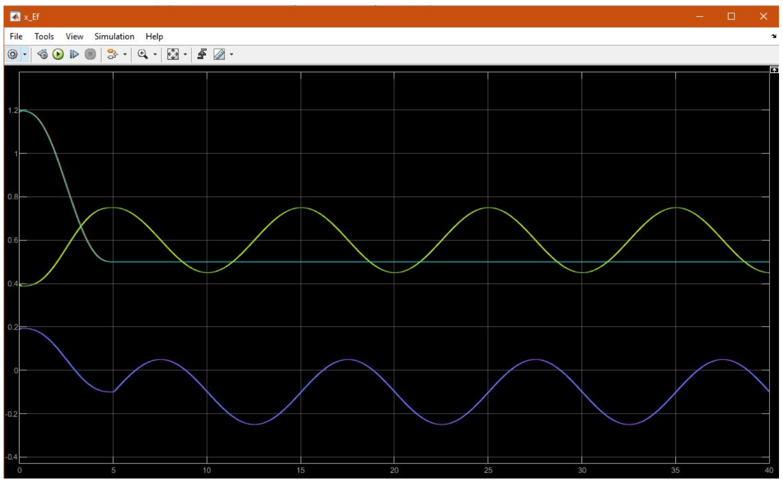

Figure 10: Plot showing the x_{ef} of the robot.

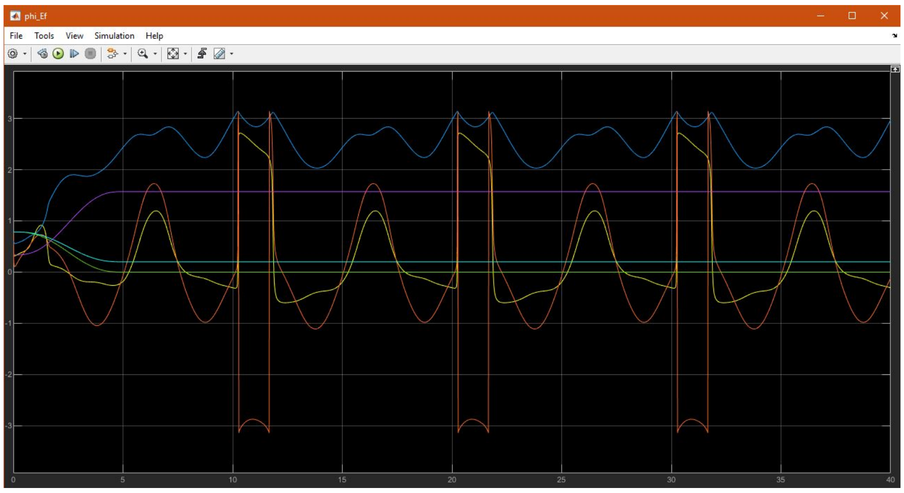

Figure 11: Plot showing the ϕ_{ef} of the robot.

We then tried without the null-space we got the following results (12):

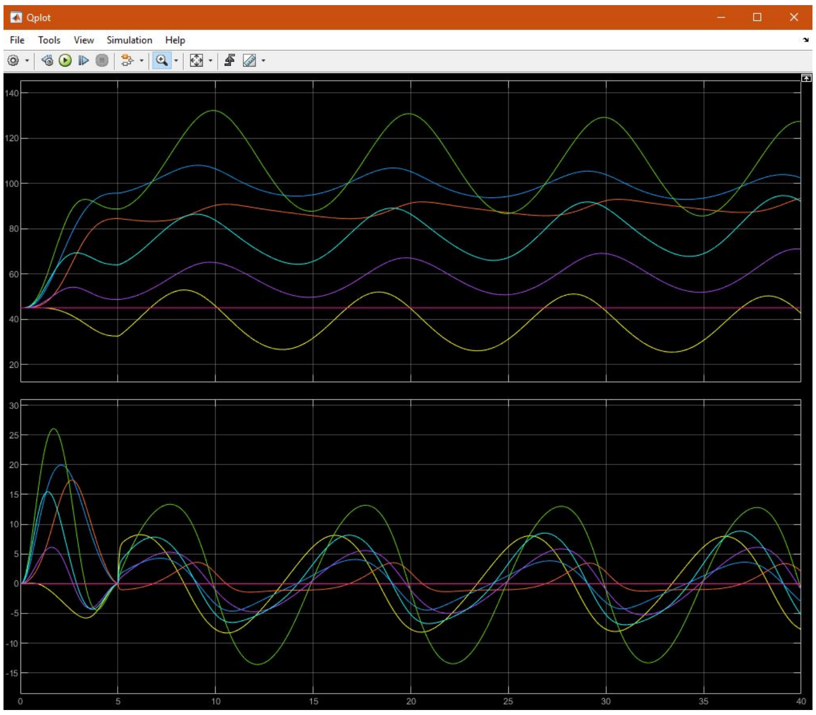

Figure 12: Plots showing the positions of the joints of the robot without null-space.

When instead using 6 degrees of freedom we get the following results 13:

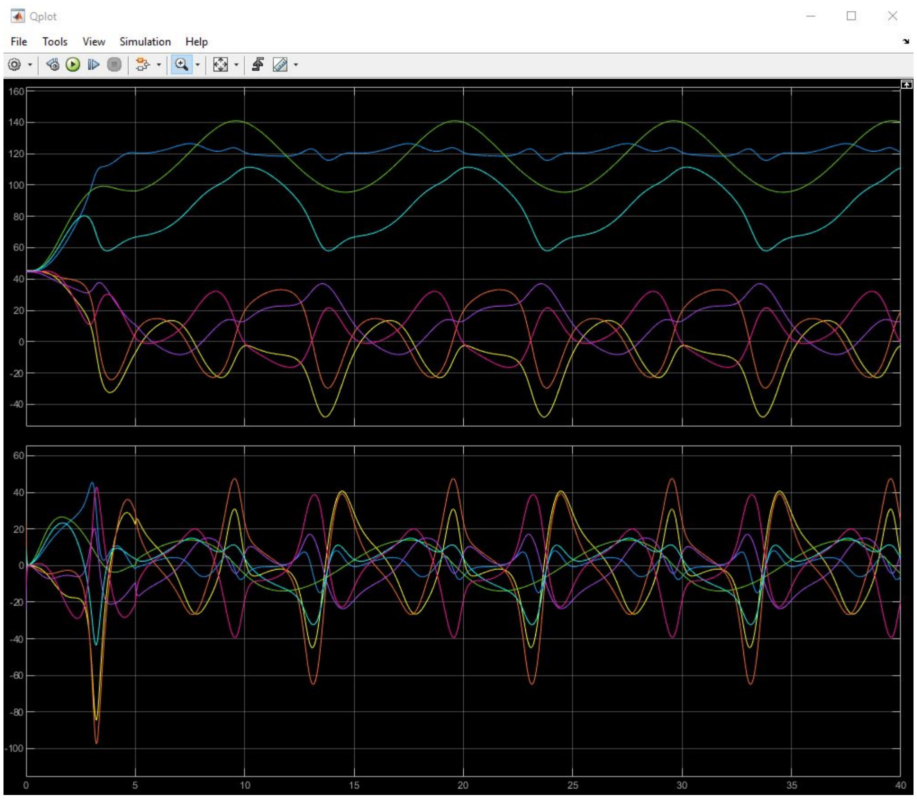

Figure 13: Plots showing the positions of the joints of the robot with 6 DOF's isntead.

(To note: All trials is based on the first trials with all its parameters. When changing something that is the base to compare to.) Firstly we can conclude that the curves with a null-space i.e. performing the secondary task gives the curves a more chaotic look since the secondary task is to oscillate all the joints. When not using the the null-space we can see that 1 of the joints is totally still and linear while the other smoothly oscillates to perform the first task. If we look at the other trial instead we can see that using 6 degrees of freedom is more chaotic than using 3. The robot is using either just the 3 first m's which is just position and in the second trial with 6 degrees of freedom we use all of the m's which is both the positions and rotations. The primary task for the robot is to control the amount of defined m's. And since we have this primary task when defining that we want to use 6 degrees of freedom and the primary task is to use them all we can see the more chaotic results since the robot have to move in both position and rotation.

And with 3 degrees of freedom just in position. When changing to static the plot of x_{ef} also is static. To also note is that the initial position impacts how to robot perform in the beginning. Putting it in a "bad" position according to the task can make it take unnecessary or complicated routes. That's why the curves are not oscillating smoothly in the beginning when the robot is on its way towards starting performing the task.