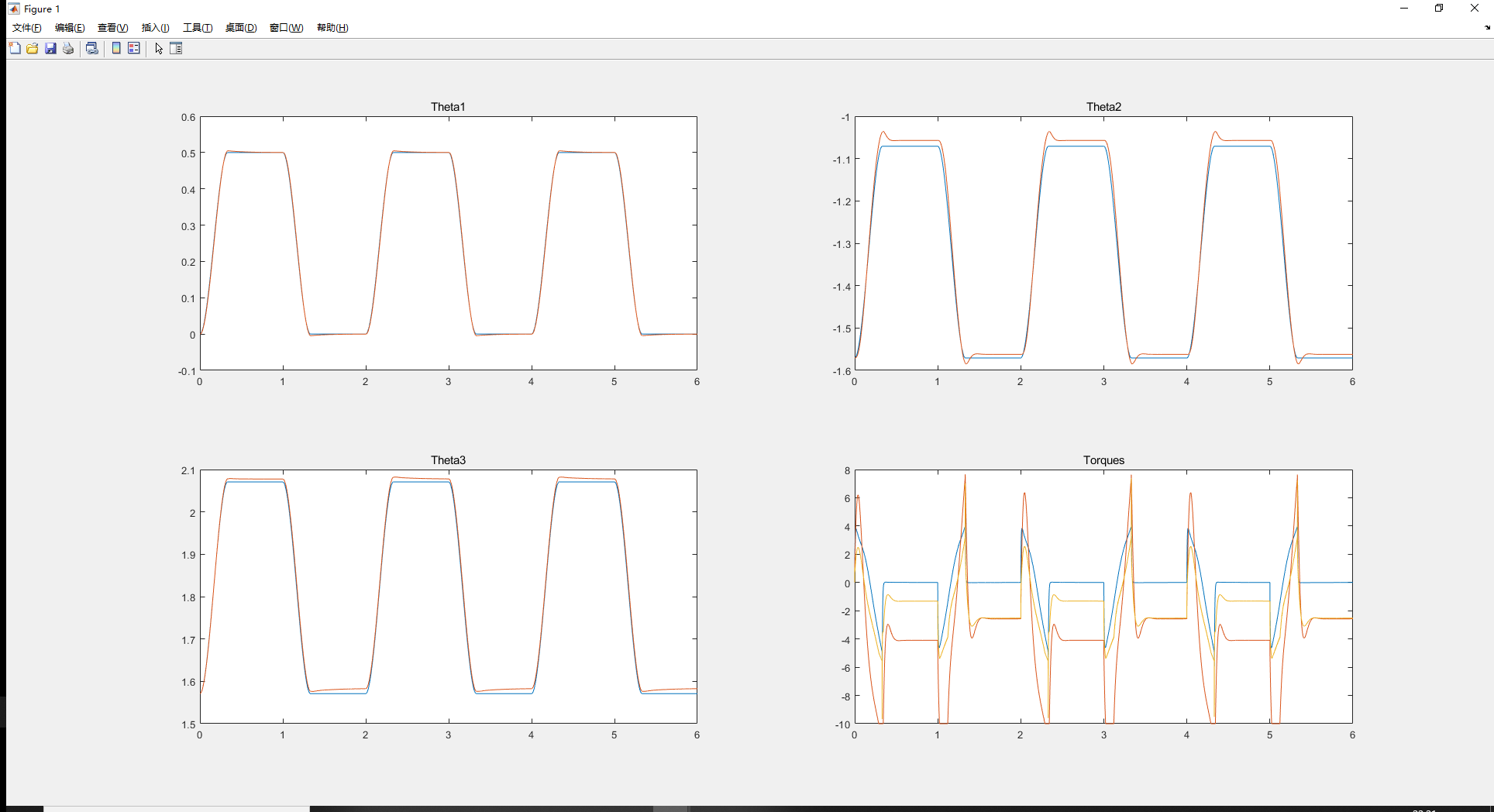
Jiaming

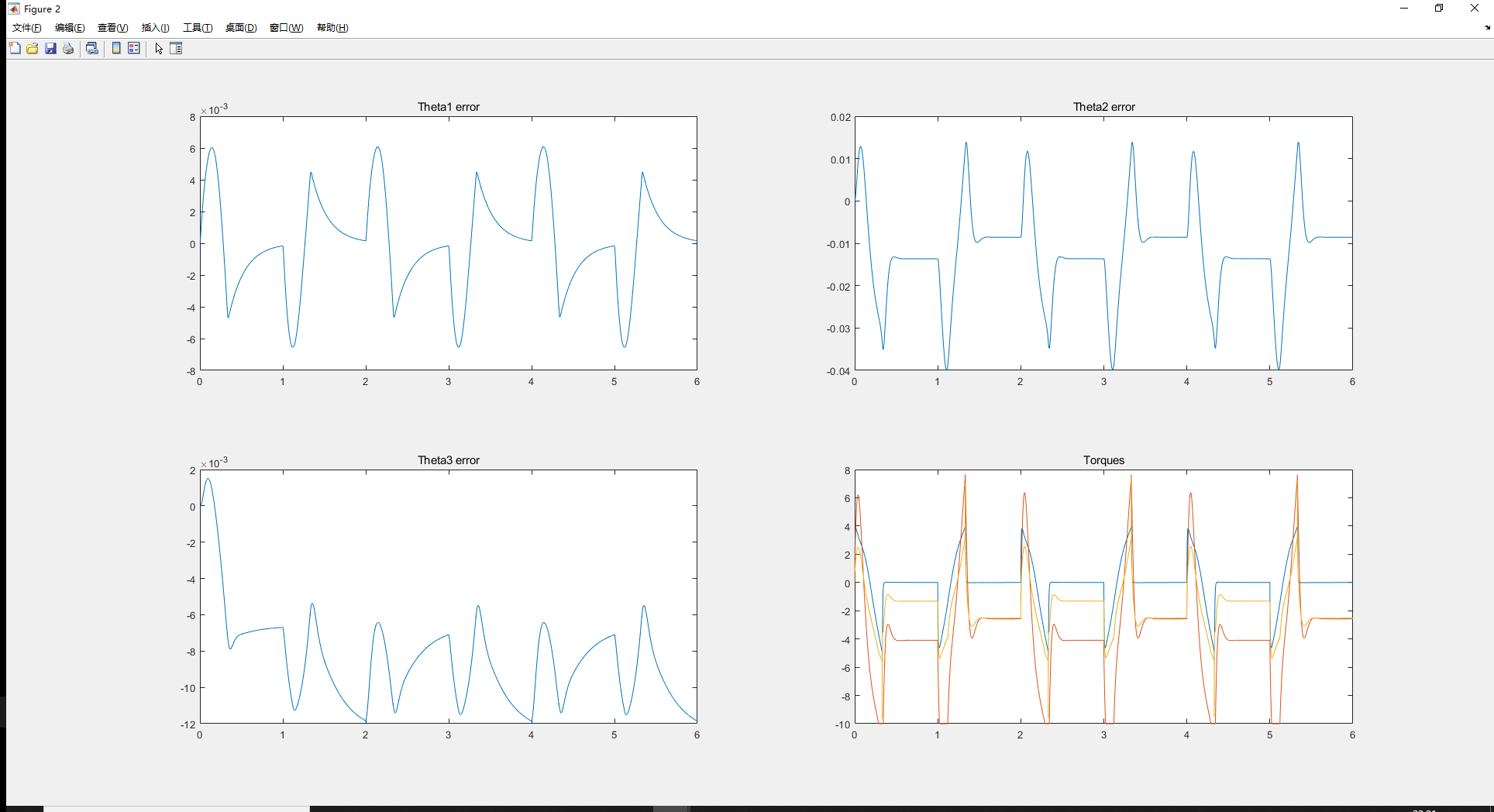
1. Inverse Dynamics Control Algorithm

To control the robot joints move to the desired angle, we need to know the desired torque to drive the robot in the real lab or the dynamics model in the simulation. To get the torque, we have to know the dynamic equation of the system, which can calculate the torque given the vector of the acceleration of each joint. This is also called the inner loop of this system. To get the acceleration, it is not a good idea to use the second derivative of the angle return from the encoder, because the error will be very large. So for the outer loop, we can give the desired angle, angular velocity and acceleration as input. Using the feedforward control to generate the acceleration, and that will be the bridge between the outer loop and the inner loop.

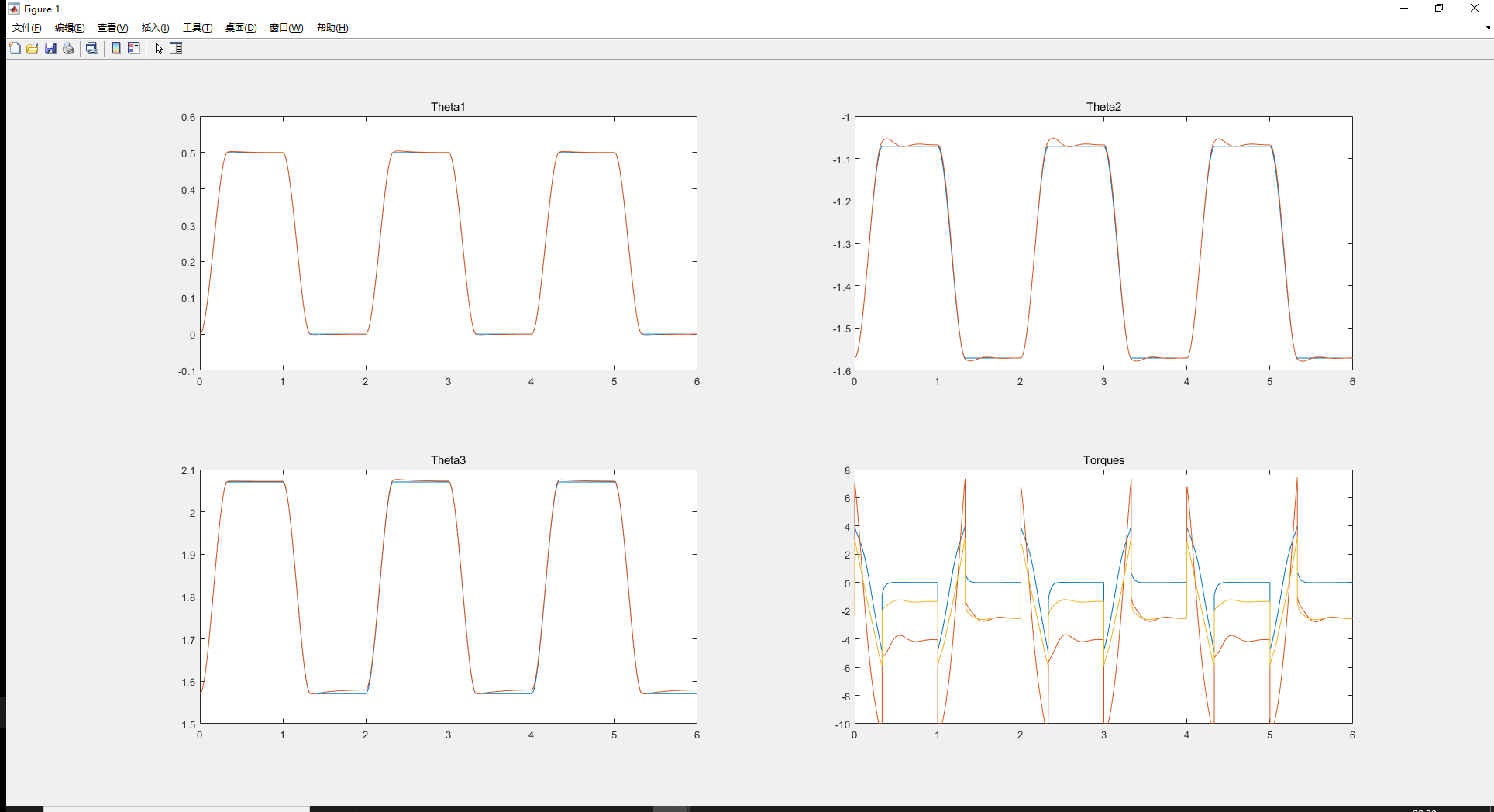
1. Add mass control performance

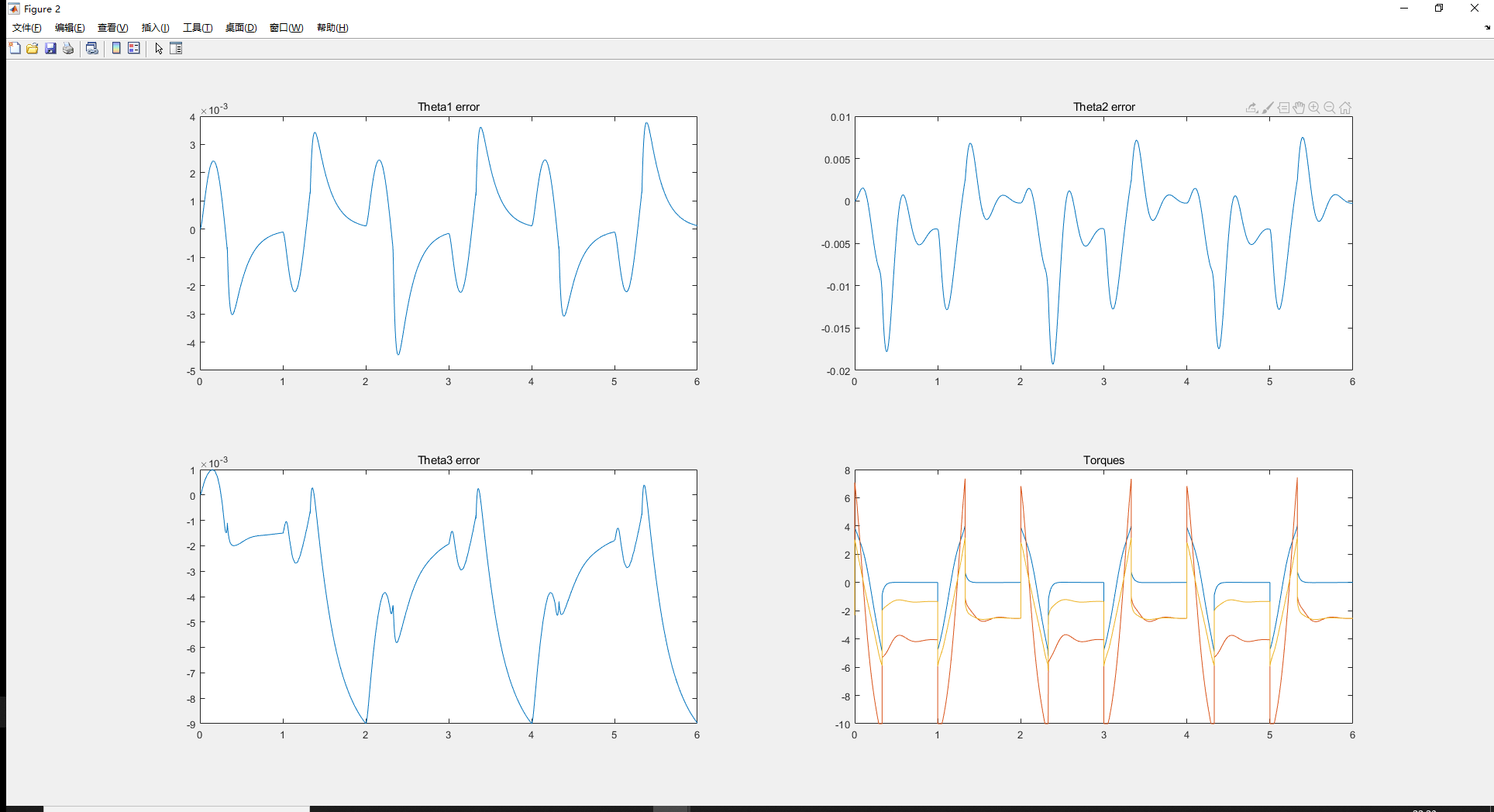
Task 6: Joint Space PD plus Feedforward + Added Mass





Task7: Inverse Dynamics control + Added Mass





1. Comparison

As the same parameter we use in the task 4 and task 5, when we add mass, the performance is worse. Among which the PD + feedforward control performances much worse, especially the steady error is larger. And when we add mass it will larger the nonlinear part, so if we linearize the system, the error will be larger too. As a conclusion the inverse dynamics control is more robust. And if we totally don’t know the parameter of the robot, we need to do the system specification first. The PD + feedforward control is less effective than the inverse dynamics control.

Note: Last time my add mass controller doesn’t work is because I didn’t add the p.ctrl\_flag = 6/7 condition in the controller code, I wrongly thought the starter code has already included it in other places. So last time my task 6/7 do not have a controller, so the performance is unsteady