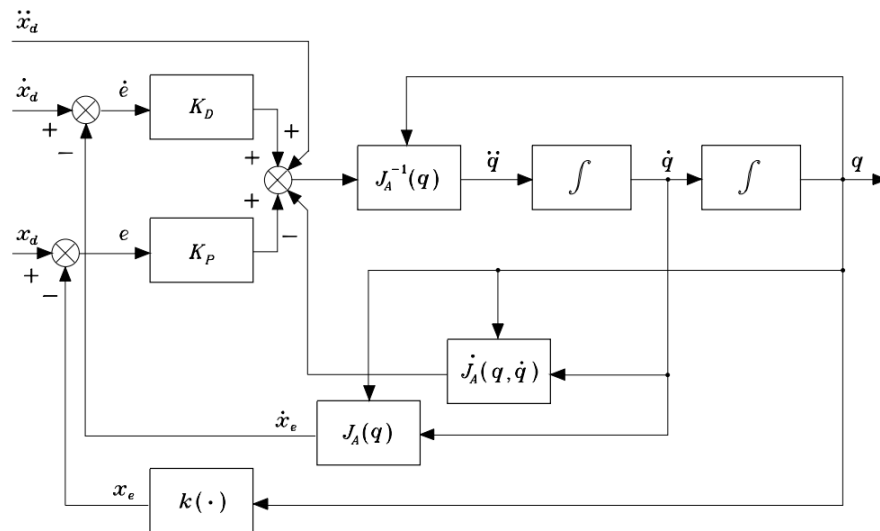ht2383   N15040025   Hua Wen, Tsuei
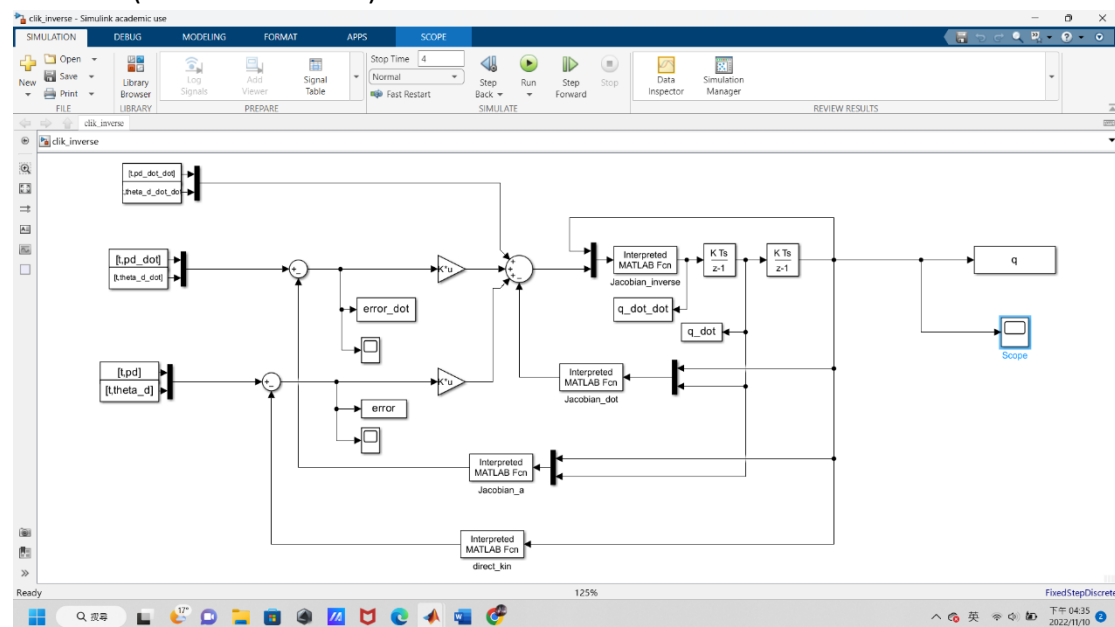
Project2 report

Part1:

142   3 Differential Kinematics and Statics

Fig. 3.14. Block scheme of the second-order inverse kinematics algorithm with Jacobian inverse
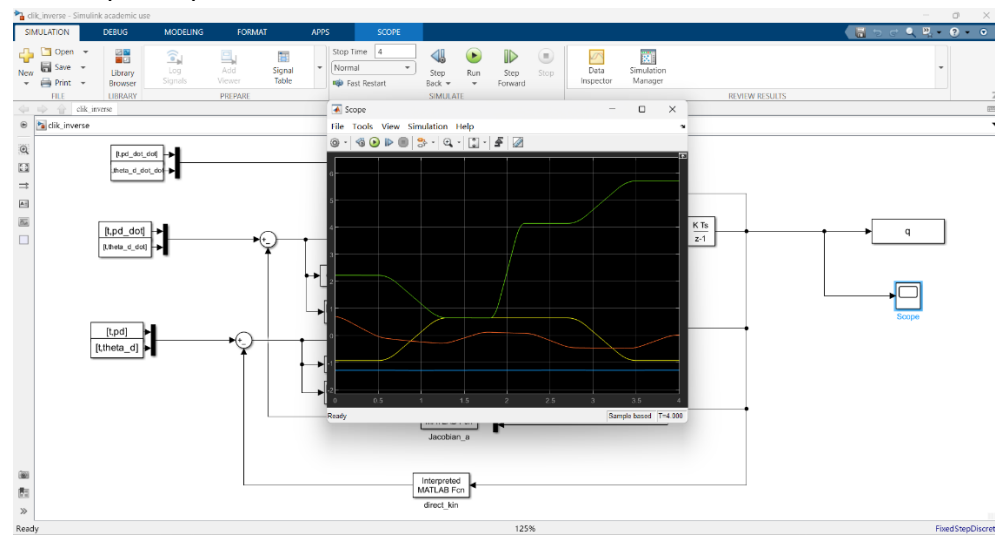
According to the scheme on textbook, I draw the same structure on matlab using Simulink.(run the init.m file)
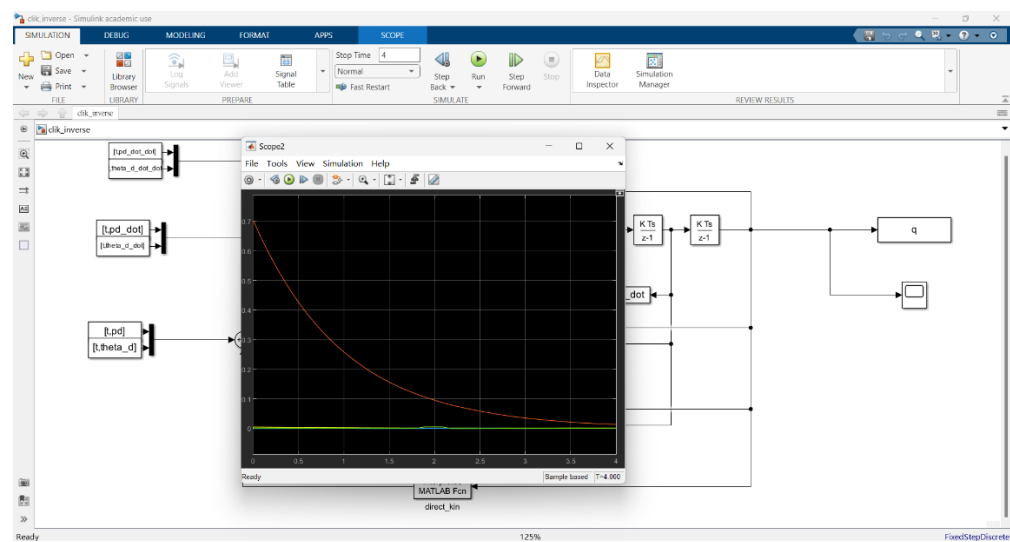

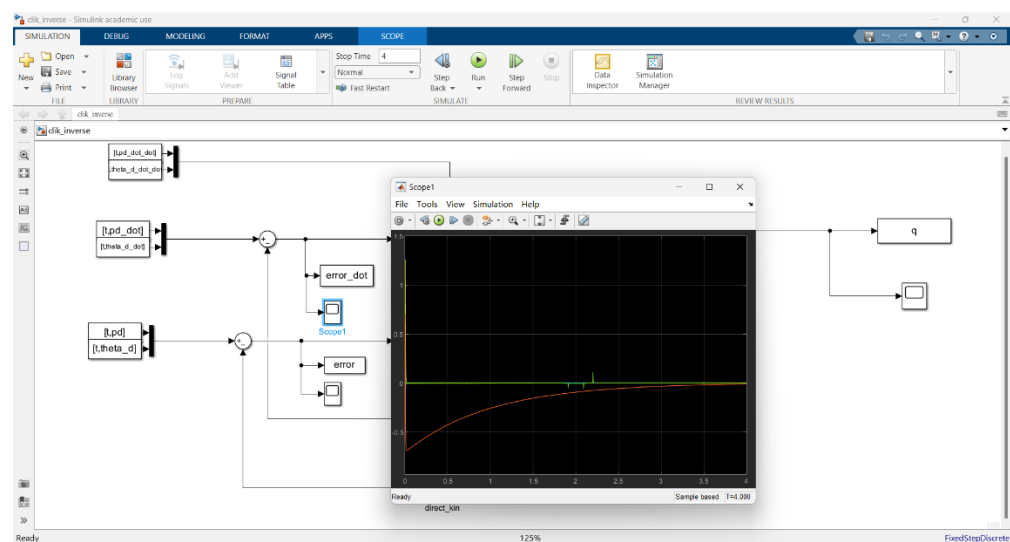For KD KP I set them both 300.

Put scope on e, e_dot & q to observe

the scope of q:

the scope of error:

the scope of error_dot:

Analytic Jacobian:

```
-a1*sin(th1)-a2*sin(th1+th2)        -a2*sin(th1+th2)        0        0
a1*cos(th1)+a2*cos(th1+th2)         a2*cos(th1+th2)         0        0
0                                   0                       -1       0
1                                   1                       0        -1
```

Jacobian inverse:

```
-a1*sin(th1)-a2*sin(th1+th2)        -a2*sin(th1+th2)        0        0
a1*cos(th1)+a2*cos(th1+th2)         a2*cos(th1+th2)         0        0
0                                   0                       -1       0
1                                   1                       0        -1
```

Jacobian dot:

```
-a1*sin(th1)-a2*sin(th1+th2)        -a2*sin(th1+th2)        0        0
a1*cos(th1)+a2*cos(th1+th2)         a2*cos(th1+th2)         0        0
0                                   0                       -1       0
1                                   1                       0        -1
```
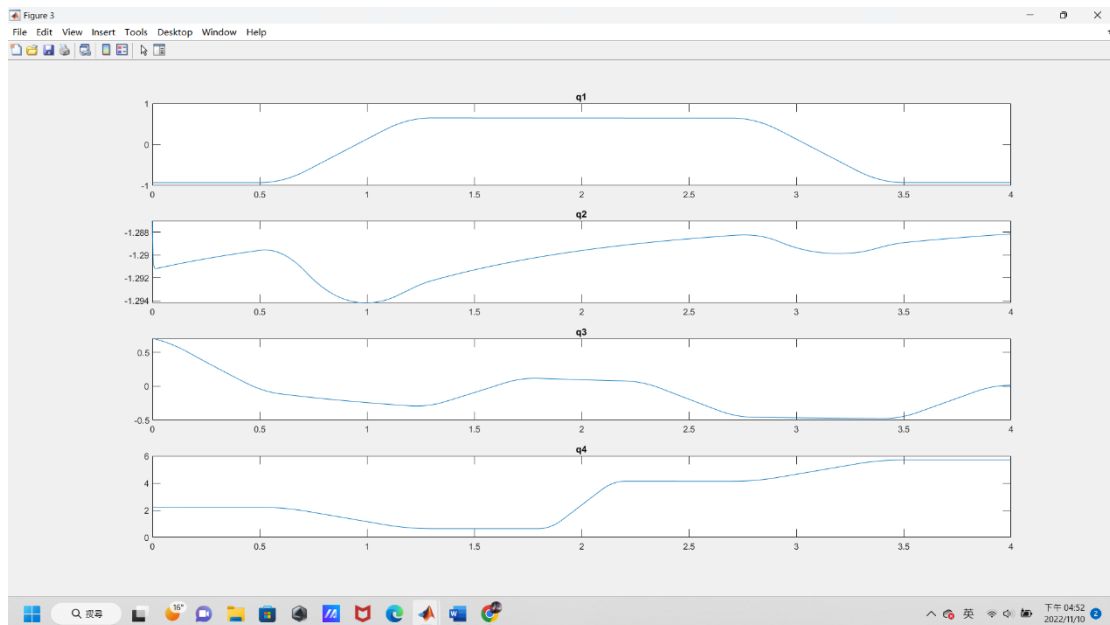
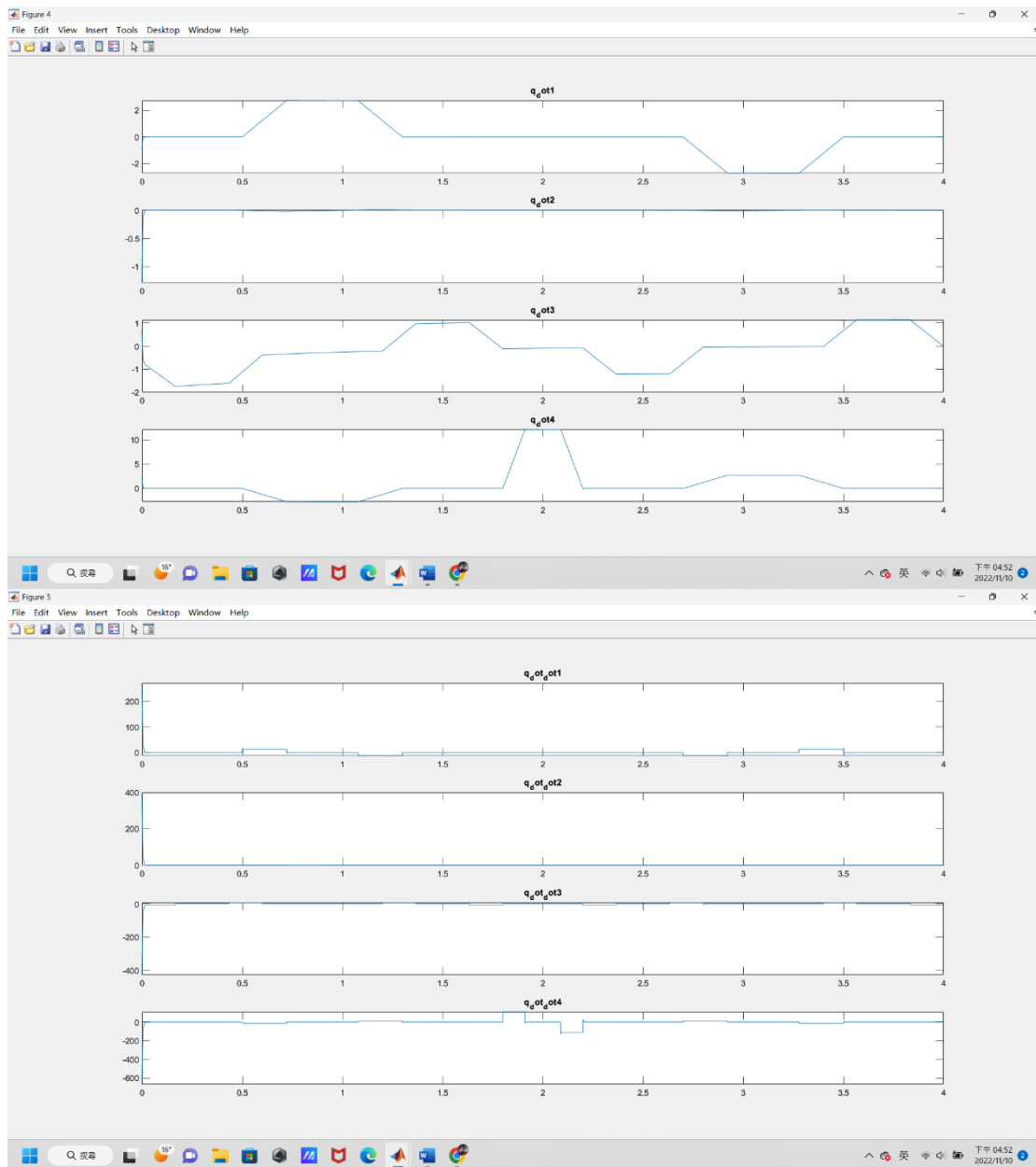If you run the plot_output.m file you will get all the Joint value and Operational space Error.

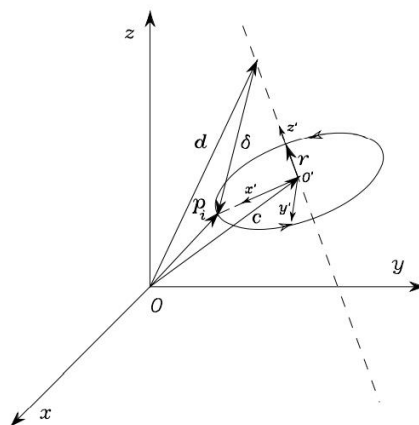Operational space Error:
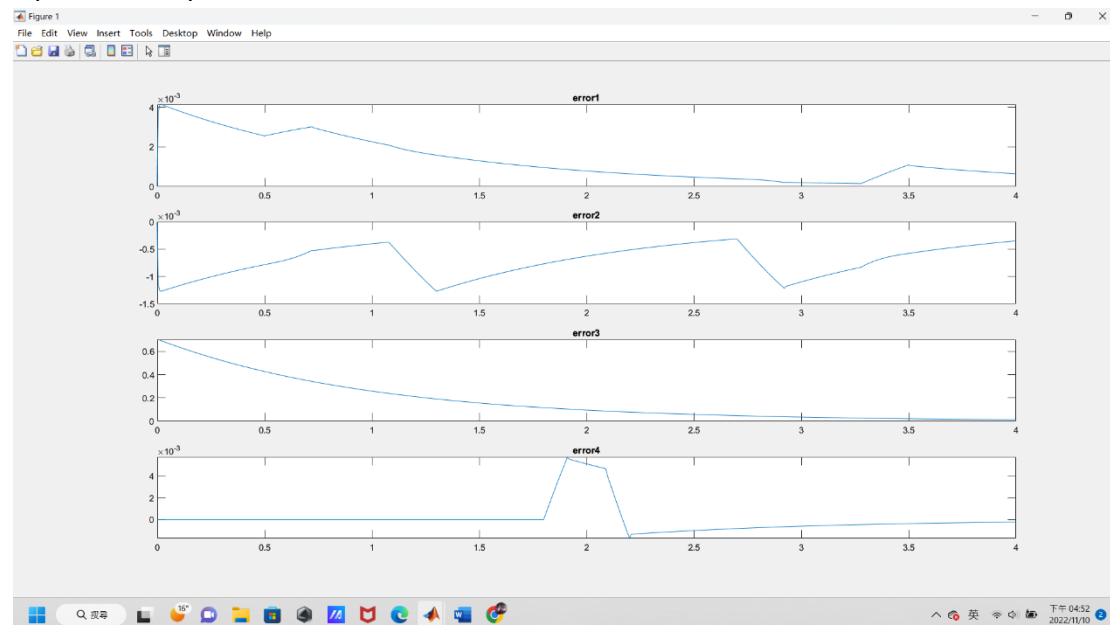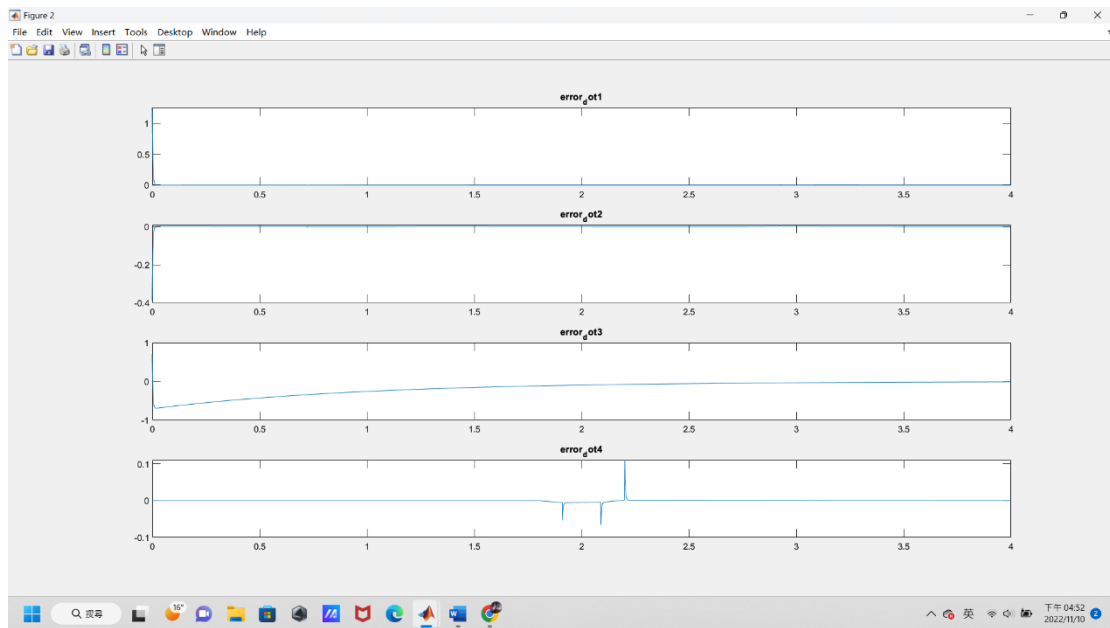
Joint value:

Part2:

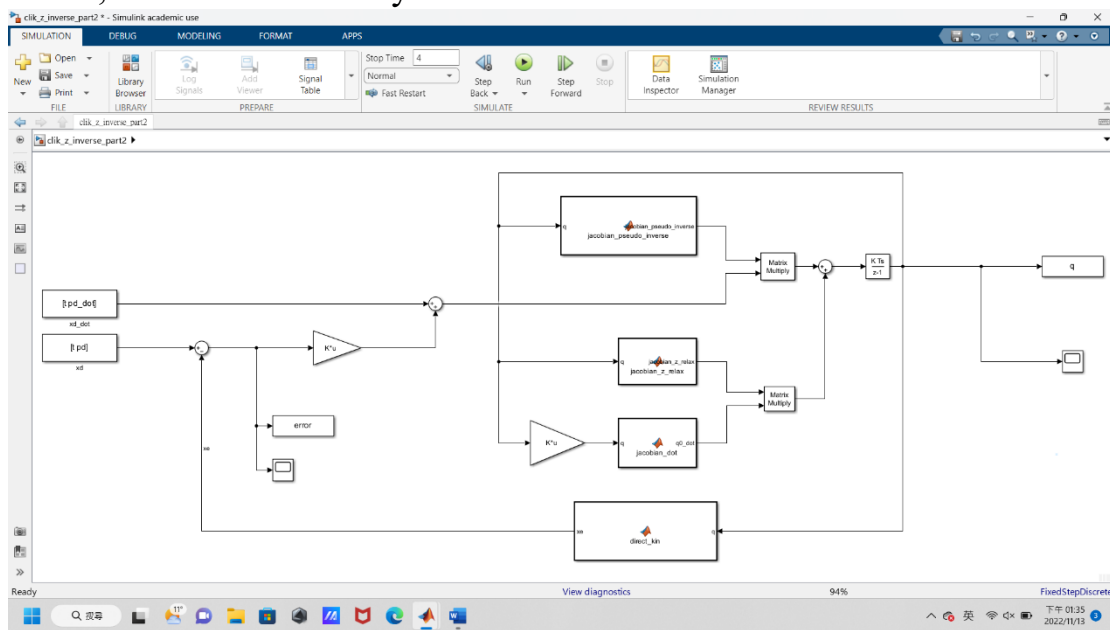**Fig. 4.12.** Parametric representation of a circle in space

$$q(t) = \begin{cases} q_i + \frac{1}{2}\ddot{q}_c t^2 & 0 \le t \le t_c \\ q_i + \ddot{q}_c t_c \left(t - t_c/2\right) & t_c < t \le t_f - t_c \\ q_f - \frac{1}{2}\ddot{q}_c \left(t_f - t\right)^2 & t_f - t_c < t \le t_f \end{cases} \qquad (4.8)$$

$$\dot{p}_e = R \begin{bmatrix} -\dot{s}\sin\left(s/\rho\right) \\ \dot{s}\sin\left(s/\rho\right) \\ 0 \end{bmatrix} \qquad (4.44)$$
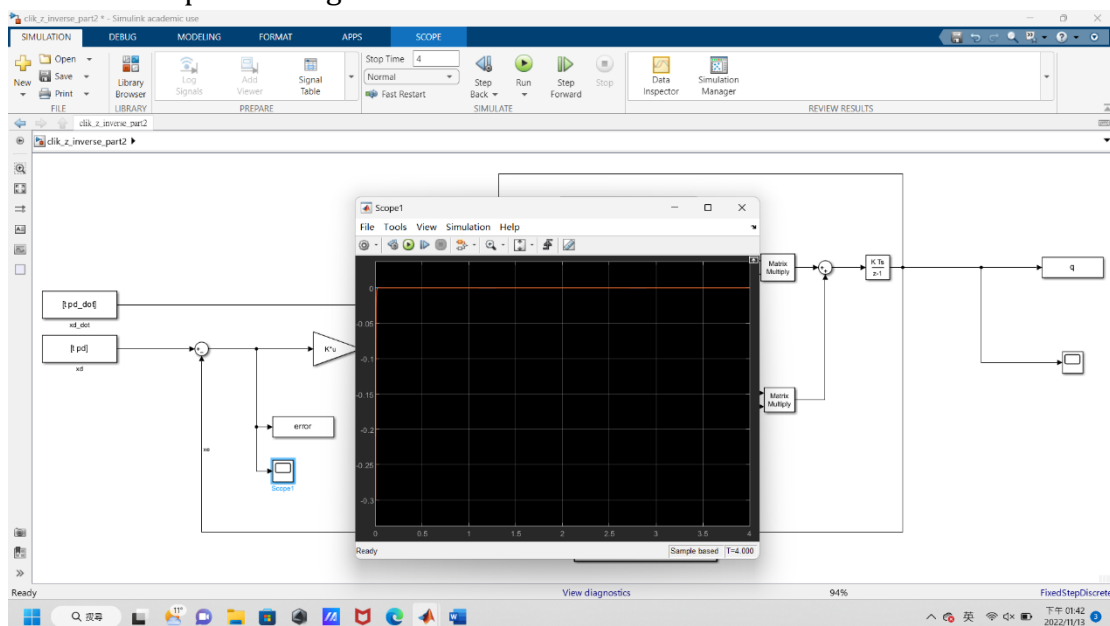
There should only have 3 rows in jacobian pseudo-inverse is given by:
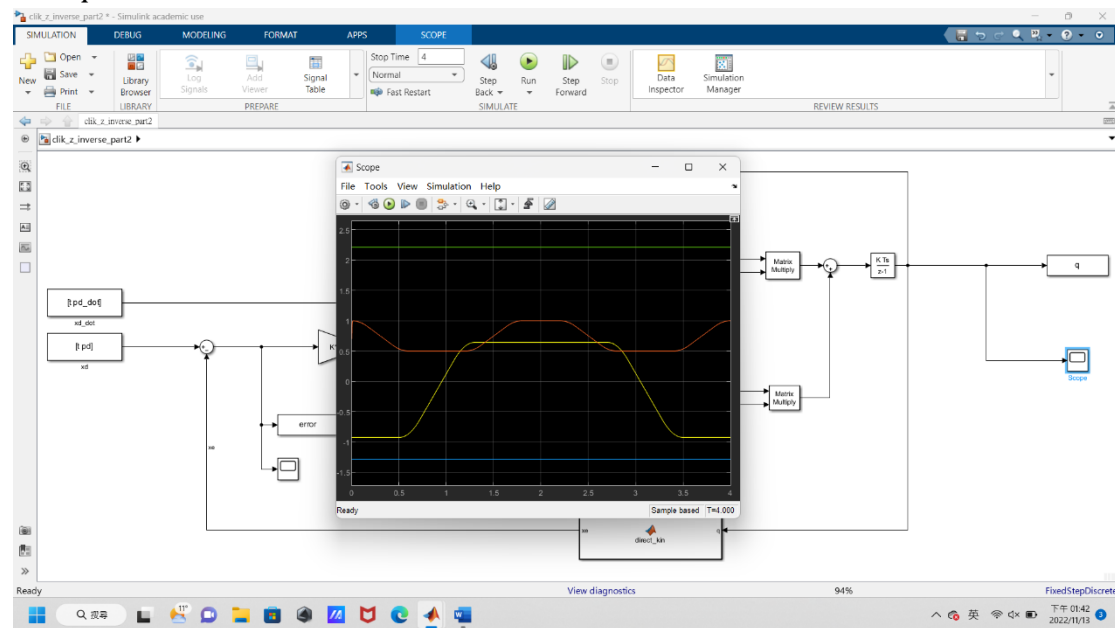$$J^\dagger = J^T (J J^T)^{-1}$$
according to the equation I think the Simulink scheme should look like below, this time I write my code inside Simulink function:

From the scope we can get error:

and q:

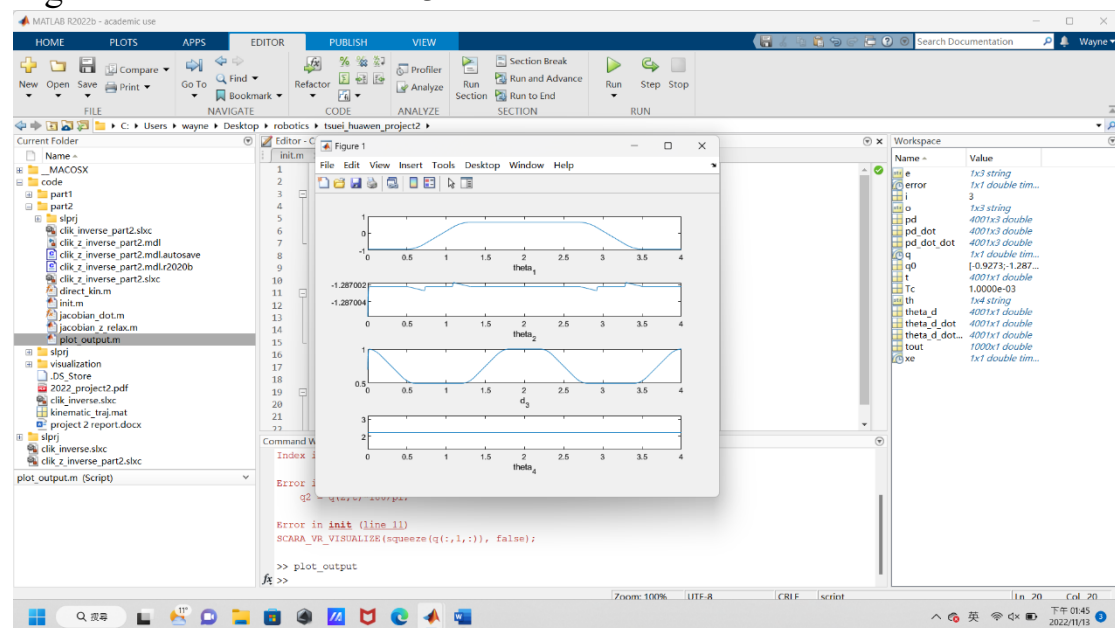To get the Joint variables and errors in the operational space(please run the plot_output.m file):

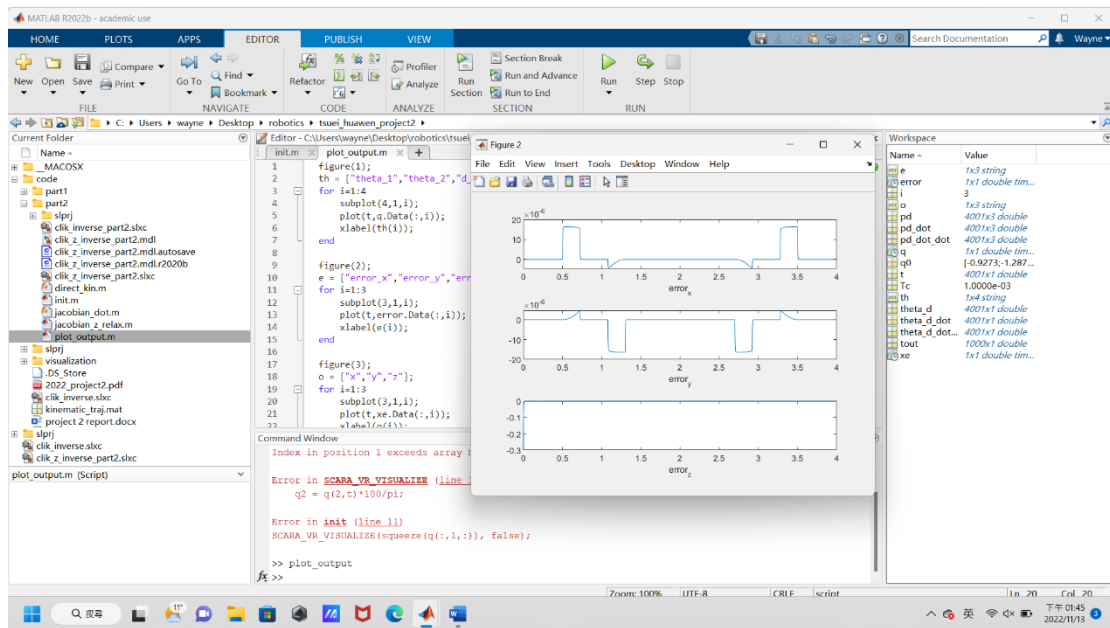Figure1: for theta1 theta2 d3 theta4

Figure2: for the error

Figure3: for x y z