Final Project controller and results

Part A. Controller design

Note1: for best understanding of this Readme file and the whole project, please open the MATLAB code since I will be using the same names for the variables throughout the project, thank you.

Note2: Throughout the MATLAB code, capital letter T will be used for the robot configuration and X will be used for the end-effector.

STEP 1: Generate reference trajectory

With the given initial configurations of the robot and end-effector, we could plug in `TrajectoryGenerator` function to generate the reference trajectory.

```
X_reference_traj =
TrajectoryGenerator(Xse_initial, Xsc_initial, Xsc_fianl, ...
Xce_grasp, Xce_standoff, k, a);
```

We will set an initial robot configuration and the initial end-effector will be generated in STEP 2.

```
T = zeros(1,12); % initial guess
log_robot_configuration{1,1} = [T X_reference_traj(1,13)];
```

However, the gripper state shall be the same with the reference trajectory.

STEP 2: Recursive algorithm

Since we have the reference trajectory and the initial configurations of the robot and end-effector, we could plug in these values in a three steps recursive algorithm (X update, U update, and T update)

Say for example we have 100 reference configurations, then here we will walk thorugh the first and last loop, since other loops are identical.

For the first loop,
With the initial robot configuration (`T = zeros(1,12);`) and the equation in p548, Chapter 13.5 we can do the X update, which gives us the initial end-effector configuration:

$$X(q,\theta) = T_{sb}(q)T_{b0}T_{0e}(\theta)$$

```
Tsb_q = [cos(phi) -sin(phi) 0 x; sin(phi) cos(phi) 0 y; 0 0 1 0.0963; 0 0 0
1];
Tb0 = [eye(3) [0.1662; 0; 0.0026]; 0 0 0 1];
M0e = [eye(3) [0.033; 0; 0.6546]; 0 0 0 1];
T0e = FKinBody(M0e, Blist, thetalist);

X = Tsb_q*Tb0*T0e; % X update
```

Then we use the end-effector configuration and reference trajectory and plug them in the `FeedbackControl` function to generate the commanded end-effector twist V, which we then use the Jacobian pseudoinverse to generate the controls $(u, \dot{\theta})$, this is the U update.

```
% feedback control to generate a twist for wheels and arm joints
Twistee_Integralofeorror ...
         = FeedbackControl(X_configuration, Xd_configuration,
Xd_next_configuration, Kp, Ki, timestep, Xerr_integral);
% output return Twist, Xerr, and the integral of the error
command_twist = Twistee_Integralofeorror(:,1);
log_Xerr(loop, :) = Twistee_Integralofeorror(:,2)'; % store Xerr of every
configuration
Xerr_integral = Twistee_Integralofeorror(:,3);
% Xerr_integral{loop,1} = Xerr_integral;
% turn the end-effector twist into wheel and arm joints speed
wheel_arm_joint_speeds = pinv(Je)*command_twist; % U update
```

Finally we plug the controls $(u, \dot{\theta})$ and the robot configuration into the `NextState` function and update the robot configuration to the next timestep for the second loop. This step is defined as the T update.

```
Trobot_nextstate = NextState(Trobot_currentstate, control_input, timestep,
limits);
% Trobot_currentstate = Trobot_nextstate;
    for a = 1: 12
    Trobot(1,a) = Trobot_nextstate(a,1);
    end
    Trobot(1,13) = gripper_open; % this gripper state will not be updated
    % the gripper state is directly updated in the log_robot_configuration
    % which will later be written into a csv file.

T = Trobot(:,1:12); %T update
log_robot_configuration{loop+1,1} = [T X_reference_traj(loop+1,13)]; % store
T of every configuration
```

This is the end of the first loop, and we feed the updated robot configuration back to the X update, then U date, then T update …

Finally when we enter the last loop (the 99th loop) and after the T update, we need to do one more step of X update so that the end-effector configuration could be controlled and reached the desire final end-effector configuration.

At the end of the program, we save three csv files, which are the reference trajectory, the robot configuration, and the error of end-effector at each configuration. A plot of the error of end-effector at each configuration will also be generated.

Part B. Results

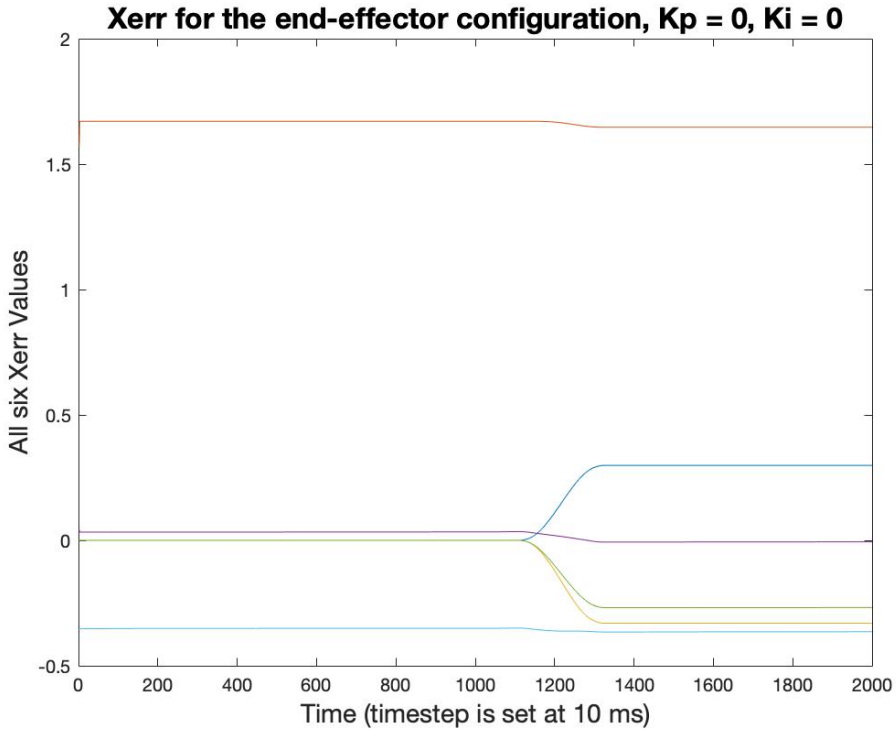
When designing the controller, not only do we have to controller the P and I gain, but also the number of cycles (time) we let the robot to operate under different cube initial and goal configurations. The performance of the robot could also be affected by the limits on wheel speeds and speed.

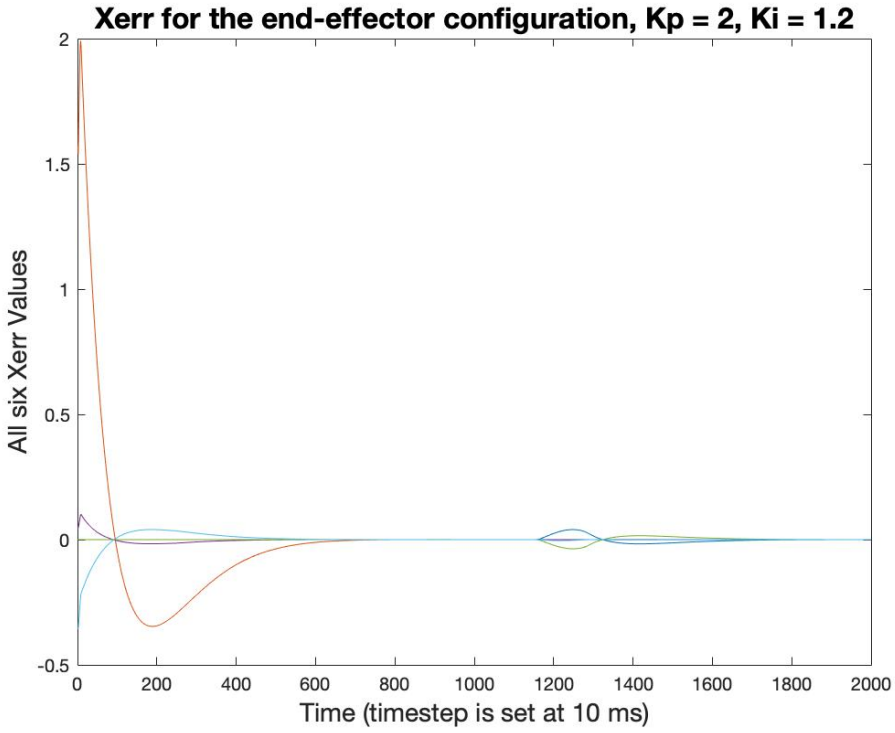
Conclusion:
From the figures below we could tell that the best performance is achieved by feedforward-plus-P, while feedforward-plus-PI often have overshoot. The first set of peaks in the figures are due to the initial end-effector configuration errors, and the second set of peaks in the figure are due to the movement between picking up the cube and placing it in the final configuration, in this period of time, the robot is not going in a straight line but a curve. Hence it is reasonable to anticipate a second set of peaks in the middle.

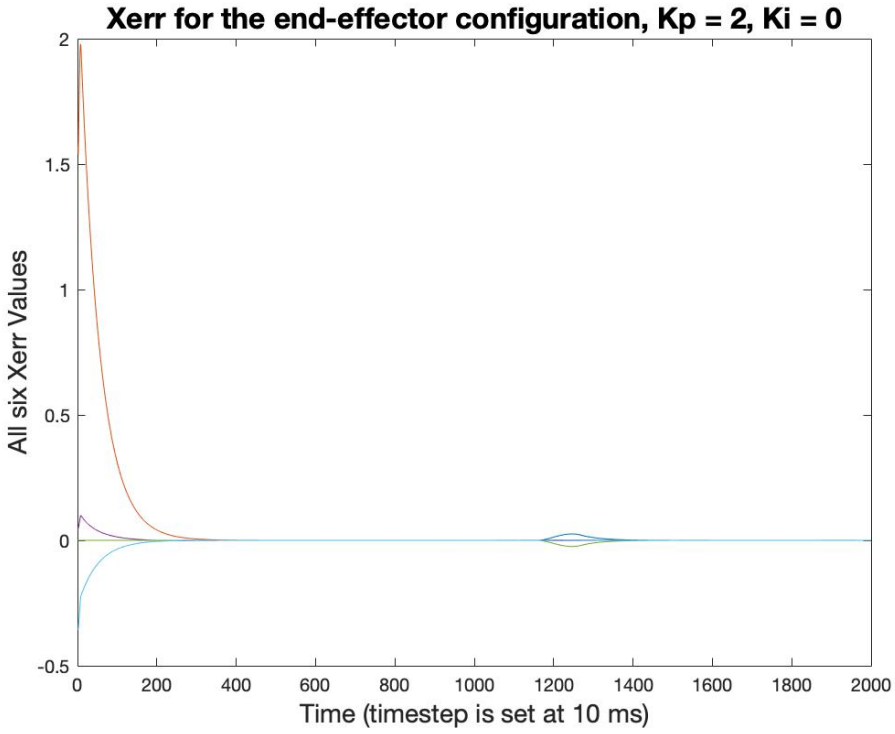
For the initial cube configurations given in Scene 6

```
Xsc_initial = [1, 0, 0, 1; 0, 1, 0, 0; 0, 0, 1, 0; 0, 0, 0, 1]; % initial
value of scene 6
Xsc_fianl = [0, 1, 0, 0; -1, 0, 0, -1; 0, 0, 1, 0; 0, 0, 0, 1]; % initial
value of scene 6
```

The following three figures represent:
(1) feedforward only (2) feedforward-plus-PI (3) feedforward-plus-P




Xerr for the end-effector configuration, Kp = 2, Ki = 1.2
All six Xerr Values
Time (timestep is set at 10 ms)
2
1.5
1
0.5
0
-0.5
0
200
400
600
800
1000
1200
1400
1600
1800
2000

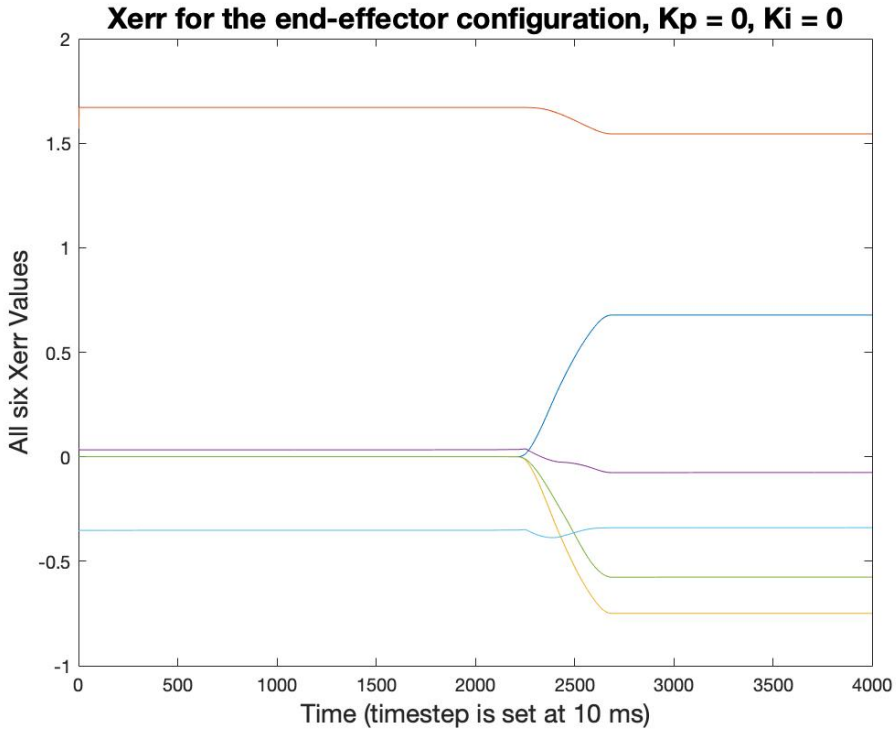

Xerr for the end-effector configuration, Kp = 2, Ki = 0
All six Xerr Values
Time (timestep is set at 10 ms)
2
1.5
1
0.5
0
-0.5
0
200
400
600
800
1000
1200
1400
1600
1800
2000

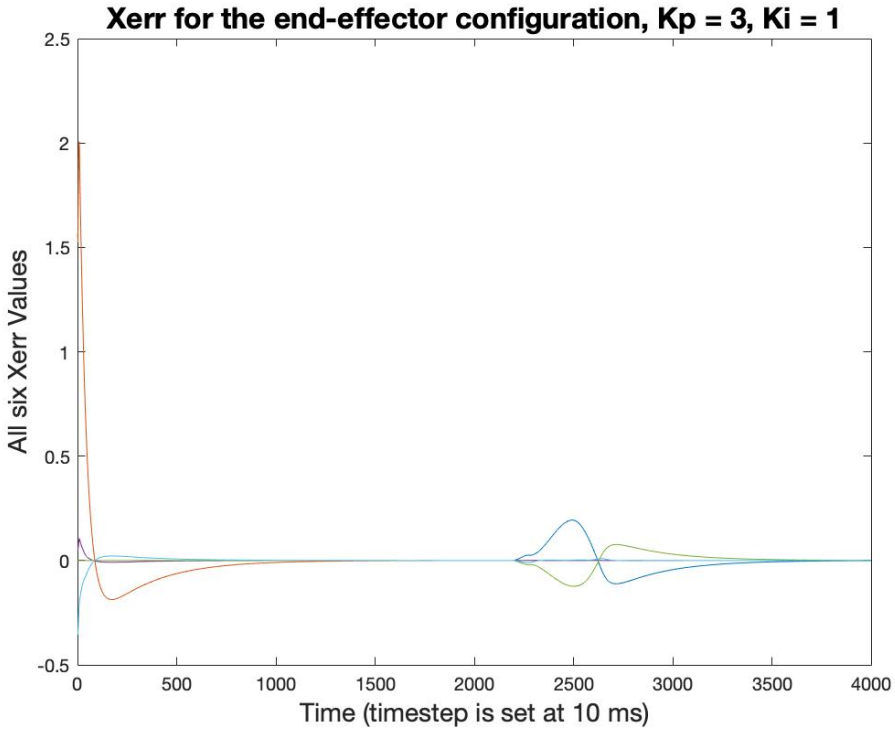
For the newTask cube configurations

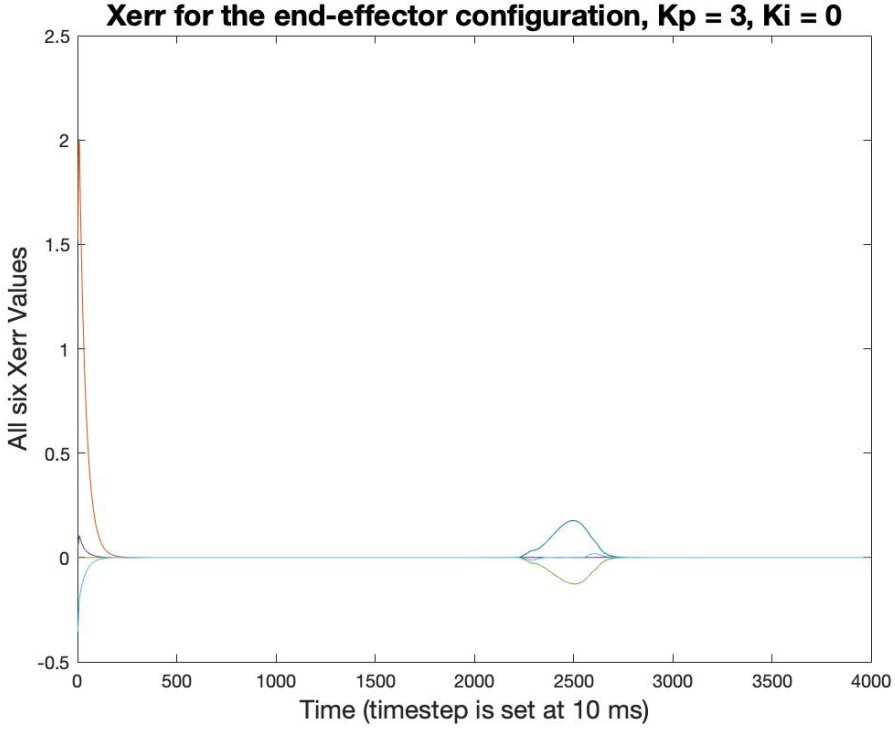
```
Xsc_initial = [1, 0, 0, 2; 0, 1, 0, 0; 0, 0, 1, 0; 0, 0, 0, 1]; % newTask
Xsc_fianl = [0, 1, 0, 0; -1, 0, 0, -2; 0, 0, 1, 0; 0, 0, 0, 1]; % newTask
```

The following three figures represent:

(1) feedforward only (2) feedforward-plus-PI (3) feedforward-plus-P




Xerr for the end-effector configuration, Kp = 3, Ki = 1
All six Xerr Values
Time (timestep is set at 10 ms)
2.5
2
1.5
1
0.5
0
-0.5
0
500
1000
1500
2000
2500
3000
3500
4000


Xerr for the end-effector configuration, Kp = 3, Ki = 0
All six Xerr Values
Time (timestep is set at 10 ms)
2.5
2
1.5
1
0.5
0
-0.5
0
500
1000
1500
2000
2500
3000
3500
4000