Final Project wrapper script

start with defining your start and end positions

```
%these are your desired perfect trajectories
clc
clear
TscI = [1 0 0 1.5; 0 1 0 0; 0 0 1 0; 0 0 0 1] %initial block config
```

```
TscI = 4×4
    1.0000         0         0    1.5000
         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000
```

```
TscF = [0 1 0  .5; -1 0 0 -.5; 0 0 1 0; 0 0 0 1] % final block config
```

```
TscF = 4×4
         0    1.0000         0    0.5000
   -1.0000         0         0   -0.5000
         0         0    1.0000         0
         0         0         0    1.0000
```

```
TseI = [0 0 1 0; 0 1 0 0; -1 0 0 .5; 0 0 0 1] % initial goal arm start position
```

```
TseI = 4×4
         0         0    1.0000         0
         0    1.0000         0         0
   -1.0000         0         0    0.5000
         0         0         0    1.0000
```

```
TceGrasp = [-.854 0 .521 0; 0 1 0 0; -.521 0 -.854 .025; 0 0 0 1] %where to grab
the block
```

```
TceGrasp = 4×4
   -0.8540         0    0.5210         0
         0    1.0000         0         0
   -0.5210         0   -0.8540    0.0250
         0         0         0    1.0000
```

```
TceStandoff = [-.854 0 .521 0; 0 1 0 0; -.521 0 -.854 .1; 0 0 0 1] %where to
standoff from the block
```

```
TceStandoff = 4×4
   -0.8540         0    0.5210         0
         0    1.0000         0         0
   -0.5210         0   -0.8540    0.1000
         0         0         0    1.0000
```

```
k=1; %This sets your number of configs per .01 sec. 2 makes it smooth in the
simulater

% this is where you are actually starting from
%starting position for  chassis phi, chassis x, chassis y, J1, J2, J3, J4, J5, W1,
W2, W3, W4, gripper state
%from sim good starting:q: 0,-.586,0, theta: 0,-.366,-.572,-.636,0
JointPos1 = [0, -.2,.2,.6,-.366,-.572,-.636,0,0,0,0,0]
```

```
JointPos1 = 1×12
         0   -0.2000    0.2000    0.6000   -0.3660   -0.5720   -0.6360         0 ···
```

Define other constants

```
Tsb= [cos(JointPos1(1)),-
sin(JointPos1(1)),0,JointPos1(2);sin(JointPos1(1)),cos(JointPos1(1)),0,JointPos1(3).
..
    ;0,0,1,.0963;0,0,0,1]; %transformation matrix that relates the fixed fram s to
the center of the base
Tb0= [1,0,0,.1662;0,1,0,0;0,0,1,.0026;0,0,0,1];%transformation matrix that relates
frame b to the start of the arm

Blist =
[0,0,0,0,0;0,-1,-1,-1,0;1,0,0,0,1;0,-.5076,-.3526,-.2176,0;.033,0,0,0,0;0,0,0,0,0];
M  =[ 1,0,0,.033;0,1,0,0;0,0,1,.6546;0,0,0,1];

XeSum=0;
dt=.01;
Kp=eye(6)*.6;
Ki=eye(6)*.01;
%store every k congig for animation
k=1;
%limits on joint and wheel velocities
limits = 100;
p=1; %counter for outputs
JointPos=[JointPos1,0];
```

do IK to find our joints are initially if needed

```
%{
eomg= .001;
ev= .0001;

T0e= TransInv(Tb0) * TransInv(Tsb) *TseI
[theta1, success] = IKinBody(Blist, M, T0e, JointPos1(4:8)', eomg, ev)
if success == 0
    error('IK failed to find a starting theta list')
end
JointPos = [JointPos1(1:3), theta1',0,0,0,0,0]

%}
```

some constants for odometry

```
r = .0475; l=.47/2; w=.3/2;
H0 = [-l-w, 1 -1; l+w 1 1; l+w 1 -1; -l-w 1 1]./r ;
F = pinv(H0);
```

```
F6 =[ 0,0,0,0;0,0,0,0;F;0,0,0,0];
```

generate trajectories

```
% set the time in seconds for each motion to occur
t= [5 1 1 1 5 1 1 1];
[outputConfigs] = TrajectoryGenerator(TseI,TscI, TscF, TceGrasp, TceStandoff, t,k);
N=size(outputConfigs,1);
```

use the feedback control function to run the robot

```
for i = 1:N-1

    %where you are now
    Tsb= [cos(JointPos(i,1)),-
sin(JointPos(i,1)),0,JointPos(i,2);sin(JointPos(i,1)),cos(JointPos(i,1)),0,JointPos(
i,3)...
    ;0,0,1,.0963;0,0,0,1]; %transformation matrix that relates the fixed fram s to
the center of the base
    T0e = FKinBody(M, Blist, JointPos(i,4:8)');
    X = Tsb*Tb0*T0e;
    %where you should be now
    Xd= TrowtoSE3(outputConfigs(i,:));
    %where you want to go next
    Xdn =TrowtoSE3(outputConfigs(i+1,:));

    [Ve, newXeSum, Xerror(:,i)] = FeedbackControl(X, Xd, Xdn, Kp, Ki, dt, XeSum);
    XeSum=newXeSum;
    %with new Ve, use jacobians to find how the joints should adjust
    Jarm = JacobianBody(Blist, JointPos(i,4:8)');
    Jbase =Adjoint(TransInv(T0e) * TransInv(Tb0))*F6;
    Je=[Jbase, Jarm];
    speeds= pinv(Je,1e-4)*Ve;    %these are u and theta dot
    [~,nextJoint,~] = NextState(JointPos(i,1:12),speeds', dt, limits);
    %check for joint limit violations
    if nextJoint(1)> .6
        Jarm(1:6,1)=[0;0;0;0;0;0];
    end
    if nextJoint(2)> .3
        Jarm(1:6,2)=[0;0;0;0;0;0];
    end
    if nextJoint(2) < -.9
        Jarm(:,2)=[0;0;0;0;0;0];
    end
    if nextJoint(3) < -2
        Jarm(:,3)=[0;0;0;0;0;0];
    end
```

```
    %recalculate Je and speeds based on updated jacobians
    Je=[Jbase, Jarm];
    speeds= pinv(Je,1e-4)*Ve;
    [nextChasis,nextJoint,nextWheel] = NextState(JointPos(i,1:12),speeds', dt,
limits);

    %the next joint positions are built from output of Nextstate, gripperposition
is passed through directly
    JointPos(i+1,:)=[nextChasis,nextJoint,nextWheel,outputConfigs(i,13)];

    %store and print every kth config and Xerr for anamation
   if rem(i,k) ==0
       outputJoints(p,:)= JointPos(i,:);
       outputXerror(:,p) = Xerror(:,i);
       p=p+1;
   end

end
writematrix(outputJoints,'outputJoints.csv')
writematrix(outputXerror','outputXerror.csv')
writematrix(outputConfigs,'outputTraj.csv')
```