TwoLinkArm

Table of Contents

This script simulates a non-planar two-link arm tracking a cubic polynomial trajectory with three different tracking controllers: inverse dynamic control, Lyapunov-based Control, and passivity-based control.

Controller Remarks

When an initial error was introduced, it can be concluded that while all three controllers still converge to the desired trajectory.

Both the Lyapunov-based controller and passivity-based controller initially converged quicker than the inverse dynamic control, but then took longer to eliminate the small remaining error.

The inverse dynamic controller did not overshoot the desired trajectory and the Lyapunov-based controller and passivity-based controller did overshoot the desired trajectory when there was some initital error.

Clean up before running anything:

```
clc, clear all, close all
```

Initial state:

```
x0 = [-0.5,0.2,0.1,0.1];
```

final state:

```
xf = [0, 0, 0, 0];
```

Final time:

```
tf = 10;
```

Option for no figure for the trajectory generator (true/false):

```
nofigure = 1;
```

Compute the cubic polynomial coefficients:

```
a1 = TwoLinkArmTraj(x0(1), x0(3), xf(1), xf(3), tf, nofigure);
a2 = TwoLinkArmTraj(x0(2), x0(4), xf(2), xf(4), tf, nofigure);
```

Set the initial error state:

```
x0_error  = [-0.6,0.4,0.15,0.05];
```

Options for plotting each controller (true/false):

```
plot_inverseDC = true;
plot_lyapunov = true;
plot_passivity = true;
```

Trajectory Generation For Plotting

Set time matrix for plotting:

```
time = 0:0.001:tf;
```

Initialize the trajectory maxtrix:

```
trajectory = zeros(length(time),2);
length(time);
for i = 1:length(time)
```

Grab the time value to use for each iteration:

```
    t = time(1,i);
```

Note *x* is in the form of *q1*, *q2*, *q1_dot*, *q2_dot*:

Cubic polynomials:

```
    vec_t = [1; t; t^2; t^3];
    theta_d = [a1'*vec_t; a2'*vec_t];
```

Save trajectory joint angles for each iteration:

```
    trajectory(i,:) = theta_d';
```

```
end
```

```
if plot_inverseDC
```

Inverse Dynamic Control

Set the tolerance options for ODE45 function:

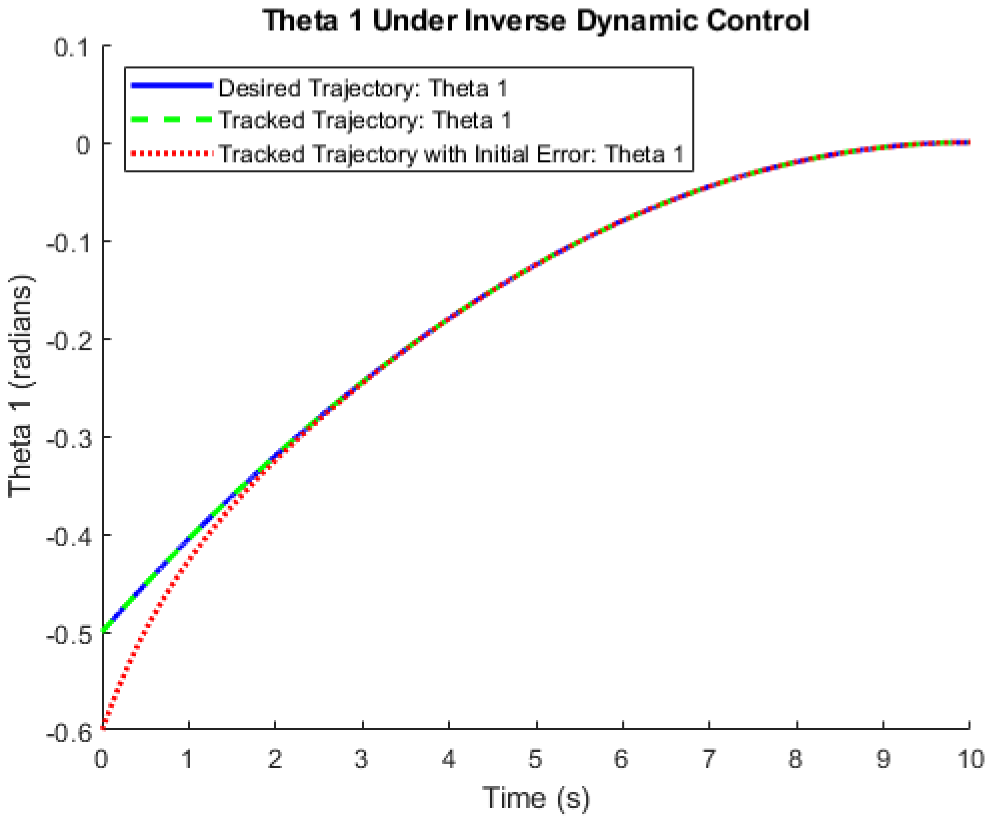
```
    options = odeset('RelTol',1e-4,'AbsTol',[1e-4, 1e-4, 1e-4, 1e-4]);
```

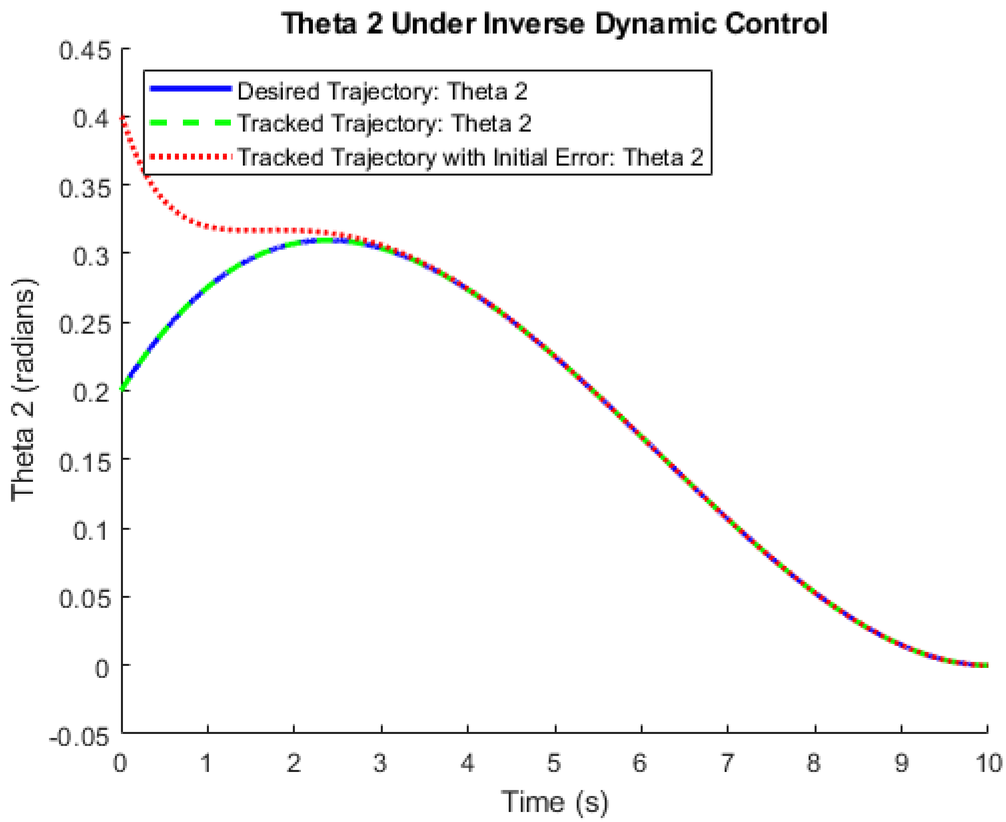
Simulate the tracking controller with no initial error:

```
    [T,X] = ode45(@(t,x) inverseDC(t, x, a1, a2), [0 tf], x0,
 options);
    [T_error,X_error] = ode45(@(t,x) inverseDC(t, x, a1, a2), [0 tf],
 x0_error, options);
```

Plot the results of the simulations:

```
   plotTrajectories(1, 'Inverse Dynamic', time, trajectory(:,1), T,
X(:,1), T_error, X_error(:,1));
   plotTrajectories(2, 'Inverse Dynamic', time, trajectory(:,2), T,
X(:,2), T_error, X_error(:,2));
```






```
end
if plot_lyapunov
```

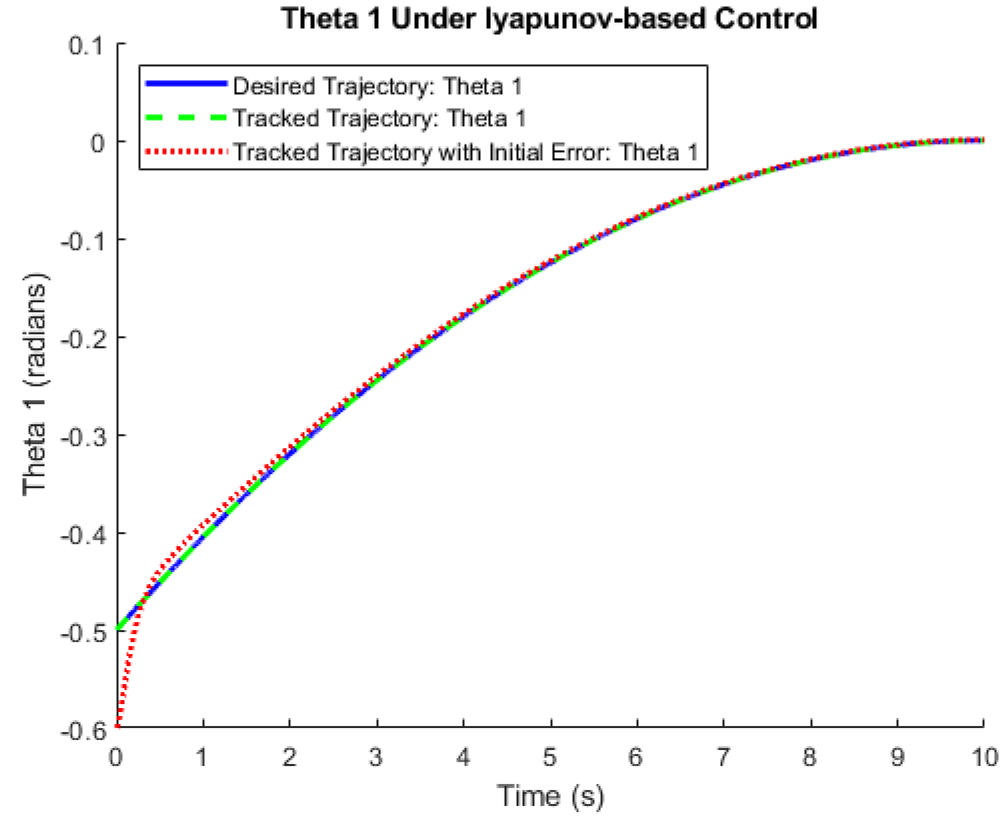
Implement the lyapunov-based control

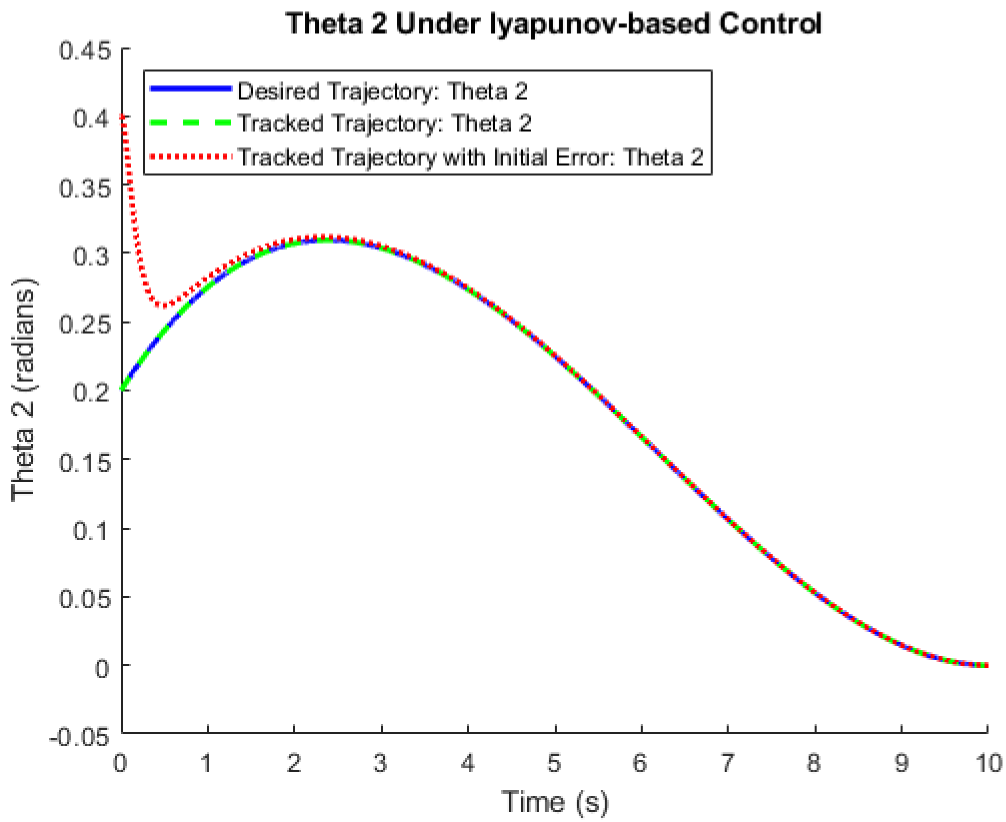
Set the tolerance options for ODE45 function:

```
    options = odeset('RelTol',1e-4,'AbsTol',[1e-4, 1e-4, 1e-4, 1e-4]);
    [T,X] = ode45(@(t,x) lyapunovCtrl(t, x, a1, a2),[0 tf],x0,
 options);
    [T_error,X_error] = ode45(@(t,x) lyapunovCtrl(t, x, a1, a2), [0
 tf], x0_error, options);
```

Plot the results of the simulations:

```
    plotTrajectories(1, 'lyapunov-based', time, trajectory(:,1), T,
 X(:,1), T_error, X_error(:,1));
    plotTrajectories(2, 'lyapunov-based', time, trajectory(:,2), T,
 X(:,2), T_error, X_error(:,2));
```


Theta 1 Under lyapunov-based Control
Desired Trajectory: Theta 1
Tracked Trajectory: Theta 1
Tracked Trajectory with Initial Error: Theta 1
Theta 1 (radians)
Time (s)
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
0
1
2
3
4
5
6
7
8
9
10


Theta 2 Under lyapunov-based Control
Desired Trajectory: Theta 2
Tracked Trajectory: Theta 2
Tracked Trajectory with Initial Error: Theta 2
Theta 2 (radians)
Time (s)
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
0
1
2
3
4
5
6
7
8
9
10

```
end

if plot_passivity
```

Implement the passivity-based control

Initialize the A matrix (joint accelerations) as a global variable:

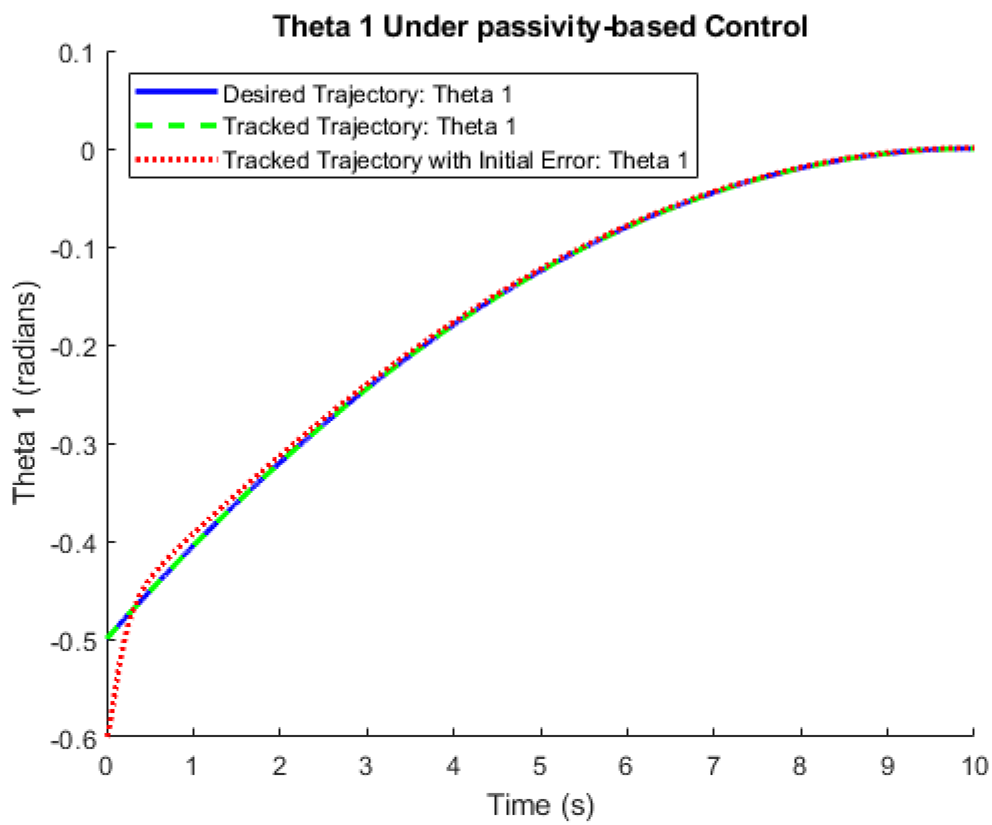
```
    global A
    A = [0;0];
```

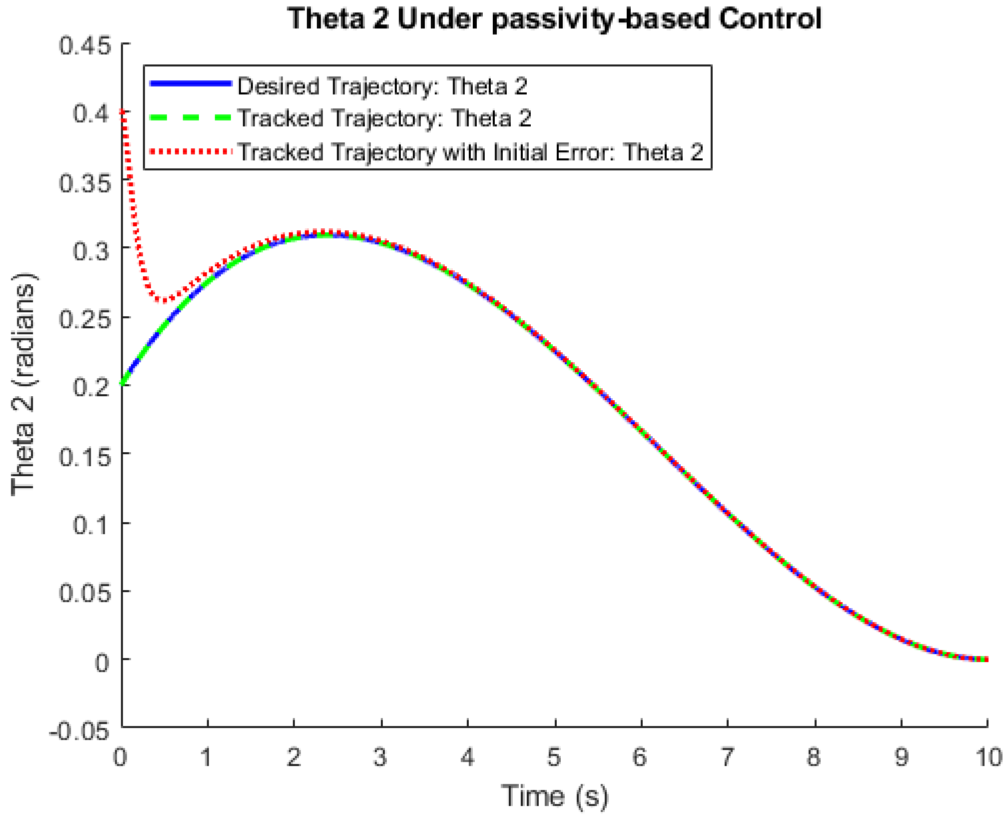
Set the tolerance options for ODE45 function:

```
    options = odeset('RelTol',1e-4,'AbsTol',[1e-4, 1e-4, 1e-4, 1e-4]);
    [T,X] = ode45(@(t,x) passivityCtrl(t,x, a1, a2),[0 tf],x0,
 options);
    [T_error,X_error] = ode45(@(t,x) passivityCtrl(t, x, a1, a2), [0
 tf], x0_error, options);
```

Plot the results of the simulations:

```
    plotTrajectories(1, 'passivity-based', time, trajectory(:,1), T,
 X(:,1), T_error, X_error(:,1));
    plotTrajectories(2, 'passivity-based', time, trajectory(:,2), T,
 X(:,2), T_error, X_error(:,2));
```






```
end
```

Published with MATLAB® R2018a