ME729 Advanced Robotics -Lab #4

2/26/2018

Sangsin Park, Ph.D.

Objectives

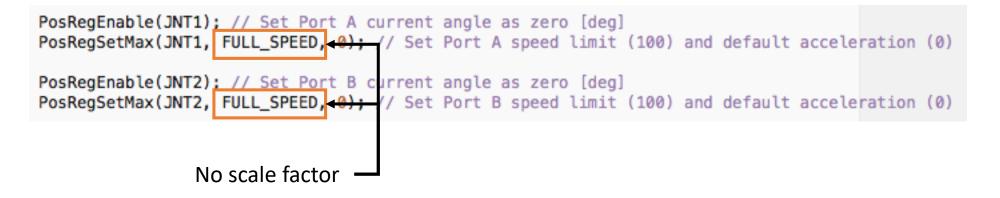
- To compare a step input with a quintic polynomial input.
- To understand a timer.

Tasks

- Complete subroutines to calculate quintic polynomials' coefficients.
- Complete quintic polynomial trajectories for joint 1 and 2

□ A step input case

- Download "Motion_Trajectories_step_input.nxc" from class web page.
- It is same as "Inv_Kine_closed_form.nxc" except for joint speed limits.
- The joint speed limits are set to 100 (full speed).
- You'll see high speed joint rotations.



A timer

- Download "Motion Trajectories timer.nxc" from class web page. ٠
- Define new constants for a timer period and a trajectory period. ٠

```
21 // Periods
22 #define timer period (0.05) // second
23 #define traj period (4.0) // second
```

Declare global variables and two functions for a timer. ٠

```
29 // Timer variables
30 unsigned long prevTick = 0;
31 unsigned long currTick = 0;
32 unsigned long timerCNT = 0;
33 float dt = 0.0;
34 float sum dt = 0.0;
```

Let's see the timer function. •



38 // for timer

39 void timer();

40 void reset timer();

- 145 **void** timer() 146 { 147 currTick = CurrentTick(); dt = (currTick - prevTick) *0.001 148 prevTick = currTick; 149 150 151 sum dt = sum dt + dt; 152 if(sum dt >= timer period) 153 154 sum dt = 0.0;155 timerCNT = timerCNT + 1; 156 } 157 }
 - Read the current system tick.
 - Calculate how much increase.
 - Save the current tick.
 - Accumulate the difference.

Since the difference is less than the timer period, we should wait for increasing a timerCNT variable until the sum of the differences reaches the timer period.

□ A timer - continued

- Let's see the reset_timer function.
- This function is to reset variables related to accumulate.

```
159 void reset_timer()
160 {
161     sum_dt = 0.0;
162     timerCNT = 0;
163 }
```

- Let's see the main function.
- Three lines are added.

```
time = timerCNT*timer_period;
time = timerCNT*timer_period;
rextOut(0, LCD_LINE6, FormatNum("Time: %.2f", time));
timer();
Calculate time in real-time by multiplying
the timerCNT by the timer period.
```

• After run this code, we can see increasing time in real-time.

□ A quintic polynomial input case

- Download "Motion_Trajectories_blanks.nxc" from class web page.
- Declare global variables and two functions for trajectories : traj1_ for joint 1 and traj2_ for joint 2.

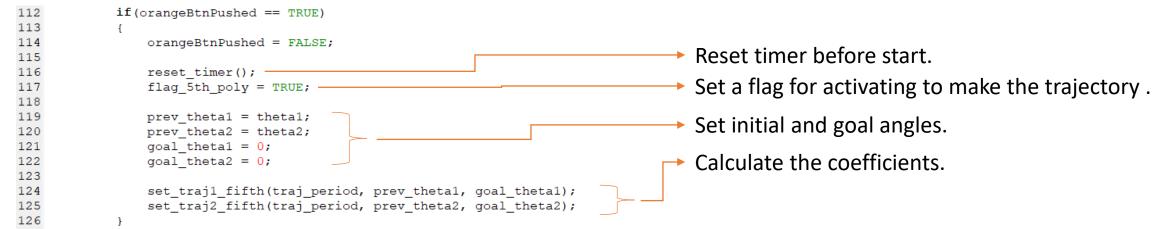
```
36 // Trajectory variables
37 float trj1_a0 = 0.0;
38 float trj1_a3 = 0.0;
39 float trj1_a4 = 0.0;
40 float trj1_a5 = 0.0;
41
42 float trj2_a0 = 0.0;
43 float trj2_a3 = 0.0;
44 float trj2_a4 = 0.0;
45 float trj2_a5 = 0.0;
```

• Those functions calculate the each polynomial's coefficients.

```
212 void set traj1 fifth(float delta t, float initial, float final)
213 {
214
       trj1 a5 = ?
       trj1 a4 = ?
215
216
       trj1 a3 = ?
217
       trj1 a0 = ?
218 }
219
                                                                              Here is your work.
220 void set traj2 fifth(float delta t, float initial, float final)
221 {
222
       trj2 a5 = ?
       trj2 a4 = ?
223
       trj2 a3 = ?
224
225
       trj2 a0 = ?
226 }
```

□ A quintic polynomial input case - continued

- We apply a quintic polynomial joint trajectory, after press buttons
- When an orange button is pressed.



• When a right arrow button is pressed : it's a same sequence.

```
130
            if (r ArrowBtnPushed == TRUE)
131
            -
132
                r ArrowBtnPushed = FALSE;
133
                IK ok = IK 2R Planar closed(-0.12, 0.12);
134
                if (IK ok == TRUE)
                                                                                         Reset timer before start.
135
136
                    reset timer();
                                                                                         Set a flag for activating to make the trajectory.
137
                    flag 5th poly = TRUE;
138
139
                    prev theta1 = theta1;
                                                                                         Set initial and goal angles.
140
                    prev theta2 = theta2;
141
                    goal theta1 = theta1 ik*gearRatio;
                                                                                         Calculate the coefficients.
142
                    goal theta2 = theta2 ik*gearRatio;
143
144
                    set_traj1_fifth(traj_period, prev_theta1, goal_theta1);
145
                    set traj2 fifth(traj period, prev theta2, goal theta2);
146
                    TextOut(0, LCD LINE6, "Solution.");
147
148
                else
149
150
                    TextOut(0, LCD LINE6, "No Solution.");
151
152
```

□ A quintic polynomial input case - continued

- If the flag is set, the quintic polynomial trajectories are generated during the trajectory period.
- In here, the trajectory period is 4.0 second.

```
154
              if(flag 5th poly == TRUE)
155
              ł
                  time = timerCNT*timer period;
156
157
                  TextOut(0, LCD LINE6, FormatNum("Time: %.2f", time));
158
                  theta1 = ?
theta2 = ?
                                  Here is your work.
159
160
161
                  if(time >= traj_period)
162
                      flag_5th_poly = FALSE;
                                                   Generate the trajectory until the trajectory period.
163
164
165
166
              }
```

• After complete your work, we can compare joint rotations along the path to step inputs.