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.

```
PosRegEnable(JNT1); // Set Port A current angle as zero [deg]
PosRegSetMax(JNT1, FULL_SPEED, 0); // Set Port A speed limit (100) and default acceleration (0)

PosRegEnable(JNT2); // Set Port B current angle as zero [deg]
PosRegSetMax(JNT2, FULL_SPEED, 0); // Set Port B speed limit (100) and default acceleration (0)
```

No scale factor

A timer

- Download “Motion_Trajectories_timer.nxc” from class web page.
- Define new constants for a timer period and a trajectory period.

```
// Periods
#define timer_period (0.05) // second
#define traj_period (4.0) // second
```

- Declare global variables and two functions for a timer.

```
// Timer variables
unsigned long prevTick = 0;
unsigned long currTick = 0;
unsigned long timerCNT = 0;
float dt = 0.0;
float sum_dt = 0.0;
```

```
// for timer
void timer();
void reset_timer();
```

- Let’s see the timer function.

```
void timer()
{
    currTick = CurrentTick();
    dt = (currTick - prevTick)*0.001;
    prevTick = currTick;

    sum_dt = sum_dt + dt;
    if(sum_dt >= timer_period)
    {
        sum_dt = 0.0;
        timerCNT = timerCNT + 1;
    }
}
```

- `*0.001`: A scale factor to convert milli-second to second.
- `currTick = CurrentTick();`: Read the current system tick.
- `dt = (currTick - prevTick)*0.001;`: Calculate how much increase.
- `prevTick = currTick;`: Save the current tick.
- `sum_dt = sum_dt + dt;`: Accumulate the difference.
- `if(sum_dt >= timer_period) { … }`: 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.

```
void reset_timer()
{
    sum_dt = 0.0;
    timerCNT = 0;
}
```

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

```
        time = timerCNT*timer_period;
        TextOut(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.

```
// Trajectory variables
float trj1_a0 = 0.0;
float trj1_a3 = 0.0;
float trj1_a4 = 0.0;
float trj1_a5 = 0.0;

float trj2_a0 = 0.0;
float trj2_a3 = 0.0;
float trj2_a4 = 0.0;
float trj2_a5 = 0.0;
```

```
// for trajectory
void set_traj1_fifth(float delta_t, float initial, float final);
void set_traj2_fifth(float delta_t, float initial, float final);
```

  - Those functions calculate the each polynomial’s coefficients.

```
void set_traj1_fifth(float delta_t, float initial, float final)
{
    trj1_a5 = ?
    trj1_a4 = ?
    trj1_a3 = ?
    trj1_a0 = ?
}

void set_traj2_fifth(float delta_t, float initial, float final)
{
    trj2_a5 = ?
    trj2_a4 = ?
    trj2_a3 = ?
    trj2_a0 = ?
}
```

Here is your work.

❑ A quintic polynomial input case - continued

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

```
if(orangeBtnPushed == TRUE)
{
    orangeBtnPushed = FALSE;

    reset_timer();
    flag_5th_poly = TRUE;

    prev_theta1 = theta1;
    prev_theta2 = theta2;
    goal_theta1 = 0;
    goal_theta2 = 0;

    set_traj1_fifth(traj_period, prev_theta1, goal_theta1);
    set_traj2_fifth(traj_period, prev_theta2, goal_theta2);
}
```

Reset timer before start.

Set a flag for activating to make the trajectory .

Set initial and goal angles.

Calculate the coefficients.

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

```
if(r_ArrowBtnPushed == TRUE)
{
    r_ArrowBtnPushed = FALSE;
    IK_ok = IK_2R_Planar_closed(-0.12, 0.12);
    if(IK_ok == TRUE)
    {
        reset_timer();
        flag_5th_poly = TRUE;

        prev_theta1 = theta1;
        prev_theta2 = theta2;
        goal_theta1 = theta1_ik*gearRatio;
        goal_theta2 = theta2_ik*gearRatio;

        set_traj1_fifth(traj_period, prev_theta1, goal_theta1);
        set_traj2_fifth(traj_period, prev_theta2, goal_theta2);
        TextOut(0, LCD_LINE6, "Solution.");
    }
    else
    {
        TextOut(0, LCD_LINE6, "No Solution.");
    }
}
```

Reset timer before start.

Set a flag for activating to make the trajectory .

Set initial and goal angles.

Calculate the coefficients.

- 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.

```
if(flag_5th_poly == TRUE)
{
    time = timerCNT*timer_period;
    TextOut(0, LCD_LINE6, FormatNum("Time: %.2f", time));

    theta1 = ?
    theta2 = ?

    if(time >= traj_period)
    {
        flag_5th_poly = FALSE;
    }
}
```

Here is your work.

Generate the trajectory until the trajectory period.

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