Intelligent Robot Studio

From Theory to Practice

Homework 2 Robot Kinematics and Motor Encoder

Group members:

- Dang Duc An
- Truong Le Quynh Hoa
- Nguyen Luong Ngoc Tran

I. Theory

Question 1: Measure the radius of the driven wheel and the distance between two driven wheels. Based on your measurements, calculate the velocities of the left and right wheels so that the robot will go along a circle with the diameter of 2 m?

- Radius r: 0.0225m
- Distance l: 0.182m
- $\frac{V_R}{V_L} = \frac{2R+l}{2R-l}$
  $\Rightarrow V_R = 1.2\, V_L$

Question 2: Given your N20 geared encoder motor, calculate the maximum number of pulses you will get when the motor spins one revolution/round? Show details of your calculations.

- Gear Ratio = 100
- Each channel receives 7 * Gear Ratio = 7*100 = 700 pulses in one revolution

Question 3: If your robot travels half of a circle with the diameter of 2 m, how many pulses will you get for the left and right wheels in the ideal condition (for example, no slippage, perfect mechanics, etc.).

If the robot travels a whole circle of 2m-diameter, the path that the left wheel (inner wheel) travels is: $C_L = 2 * \pi * \left(R - \frac{l}{2}\right) \approx 5.7m$

Length of half a circle is $\frac{C_L}{2} = 2.85m$. We need to calculate how many revolutions a wheel has to spin to travel 3.14 m in order to calculate the number of corresponding pulses.

Circumference of a wheel is $\mathrm{C} = 2 * \pi * r = 2 * 3.14 * 0.0225 = 0.1413\ m$

Number of revolutions for the left wheel to travel through 3.14m is: $\frac{2.85}{0.1413} = 20.2\ revs$

Based on the calculation in Q1, then no. of revolutions for the right wheel is $20.2 * 1.2 = 24.2\ revs$

Based on the calculation in Q2, no. of pulses we will get for the left and right wheels are about $20.2 * 700 = \mathbf{14,140}\ \boldsymbol{pulses}$ and $24.2 * 700 = \mathbf{16940}\ \boldsymbol{pulses}$, respectively.

Question 4*:

Question 4*: As discussed in the lecture, the kinematics of a differential drive robot in the continuous time domain is given by:

$$\begin{aligned}\dot{x} &= \frac{1}{2}(V_r + V_l)\cos\theta \\ \dot{y} &= \frac{1}{2}(V_r + V_l)\sin\theta \\ \dot{\theta} &= \frac{1}{l}(V_r - V_l)\end{aligned} \tag{1}$$

In practice, the robot is often controlled by a digital computer which works in the discrete time domain. Therefore, it is necessary to convert the kinematic equations to that domain. It can be done by approximating the derivative as:

$$f'(t) = \frac{f_t - f_{t-1}}{\Delta T} \tag{2}$$

where ΔT is the sampling period.

a) From equations (1) and (2), derive the kinematic equations of the robot in the discrete time domain.

$$f'(t) = \frac{f_t - f_{t-1}}{\Delta T}$$

$$\Rightarrow f_t = f'(t)\Delta T + f_{t-1}$$

Therefore:

$$\mathrm{x_t} = \dot{x}_t.\Delta T + x_{t-1} = \frac{1}{2}.(V_R + V_L).\cos\theta_{t-1}.\Delta T + x_{t-1}$$

$$y_t = \dot{y}_t.\Delta T + y_{t-1} = \frac{1}{2}.(V_R + V_L).\sin\theta_{t-1}.\Delta T + y_{t-1}$$

$$\theta_t = \dot{\theta}_t.\Delta T + \theta_{t-1} = \frac{1}{l}.(V_R - V_L).\Delta T + \theta_{t-1}$$

b) Assume that at time $t_0 = 0$, the robot's pose is (0,0,0). Find its pose at time t_1 given that the velocities of the left and right wheels are $V_l = 1$ m/s, $V_r = 2$ m/s, respectively; the sampling period is 500 ms; and the distance l between the driven wheels is 30 cm.

We have:

$(x_0, y_0, \theta_0) = (0,0,0)$

$\Delta T = 500ms = 0.5s$

V_L = 1m/s

V_R = 2m/s

l = 0.3 m

⇨ Plug into (1):

$$\dot{x_1} = \frac{1}{2}(V_R + V_L).\cos\theta_0 = \frac{1}{2}(1+2).cos\ 0 = 1.5\ m/s$$

$$\dot{y_1} = \frac{1}{2}(\mathrm{V_R} + \mathrm{V_L}).\ \sin\theta_0 = \frac{1}{2}(1+2).\ \sin 0 = 0\ \mathrm{m/s}$$

$$\dot{\theta_1} = \frac{1}{l}(V_R - V_L)\ = \frac{1}{0.3}(2-1)\ = \frac{10}{3}\mathrm{rad/s}$$

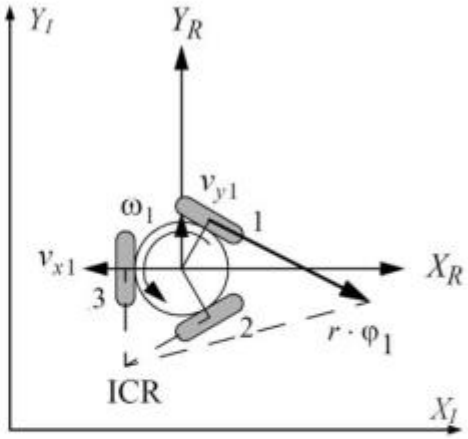
The robot's pose at time t=1:

$$\mathrm{x_1} = \dot{x_1}.\Delta T + x_0 = 1.5(0.5) + 0 = 0.75\ m$$

$$y_1 = \dot{y_1}.\Delta T + y_0 = 0\ (0.5)\ +\ 0\ =\ 0\ m$$

$$\theta_1 = \dot{\theta_1}.\Delta T + \theta_0 = \frac{10}{3}(0.5) + 0 = \frac{5}{3}\ rad$$

Question 5:



Its forward and inverse kinematics are given by:

- Forward kinematics:

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} & 0 \\ \frac{-1}{3} & \frac{-1}{3} & \frac{2}{3} \\ \frac{-1}{3l} & \frac{-1}{3l} & \frac{-1}{3l} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

- Inverse kinematics:

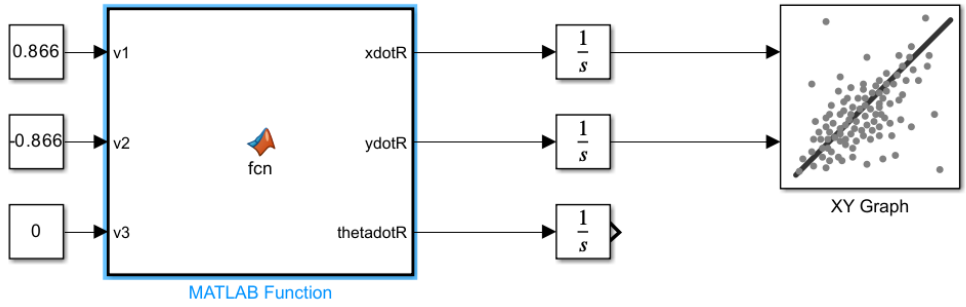
$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{-1}{2} & -l \\ \frac{-\sqrt{3}}{2} & \frac{-1}{2} & -l \\ 0 & 1 & -l \end{bmatrix}^{-1} \begin{bmatrix} V_{XR} \\ V_{yR} \\ \dot{\theta}_R \end{bmatrix}$$

where $l = 1$.

a) Using forward kinematics, build a MATLAB Simulink model of the robot.

HW2_Theory_task5 ▸ MATLAB Function

```
1  function [xdotR, ydotR, thetadotR]= fcn(v1,v2,v3)
2  l = 1;
3  xdotR = [1/sqrt(3) -1/sqrt(3) 0] * [v1;v2;v3];
4  ydotR = [-1/3 -1/3 2/3] * [v1;v2;v3];
5  thetadotR = [-1/(3*l) -1/(3*l) -1/(3*l)] * [v1;v2;v3];
6
```



b) Calculate the three wheels' velocities so that the robot is moving in the X_R direction with a velocity V_{XR} = 1m/s. Run the simulation to confirm your results.

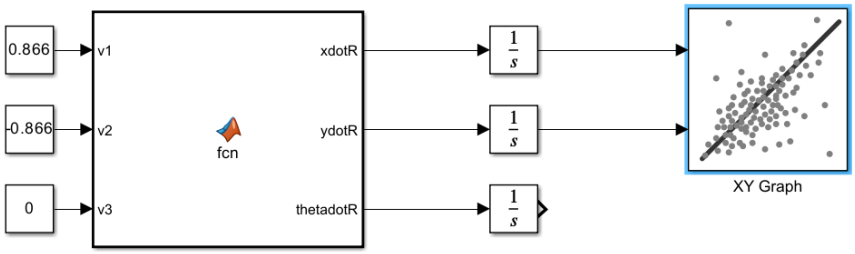
task5_theory.m *

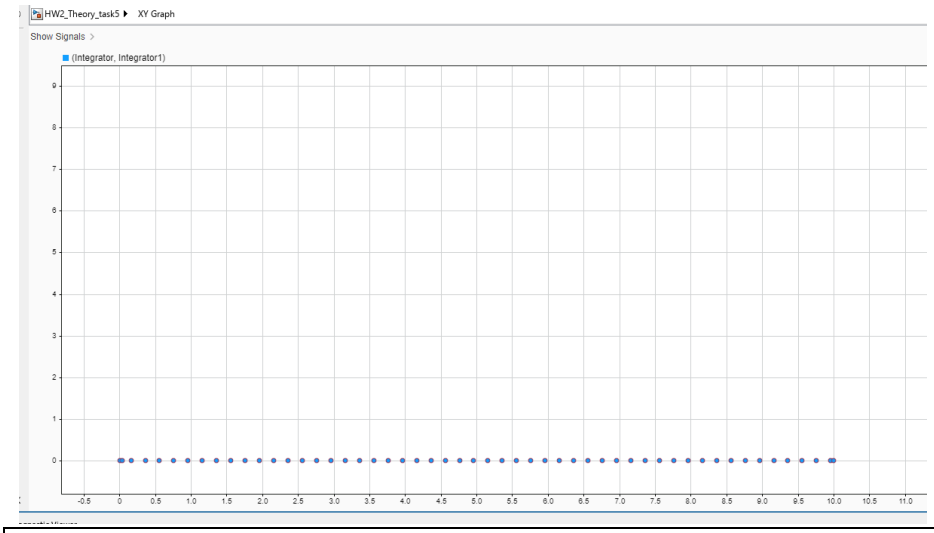
```
1   l=1
2   inverse_kinematics = [sqrt(3)/2     -1/2    -1;...
3                         -sqrt(3)/2    -1/2    -1;...
4                         0             1       -1]
5   % question_b:
6   xdotR = 1;
7   ydotR = 0;
8   thetadotR = 0;
9
10  inverse_kinematics * [xdotR; ydotR; thetadotR]
```

Command Window

```
ans =

    0.8660
   -0.8660
         0
```

Verify with Simulink:



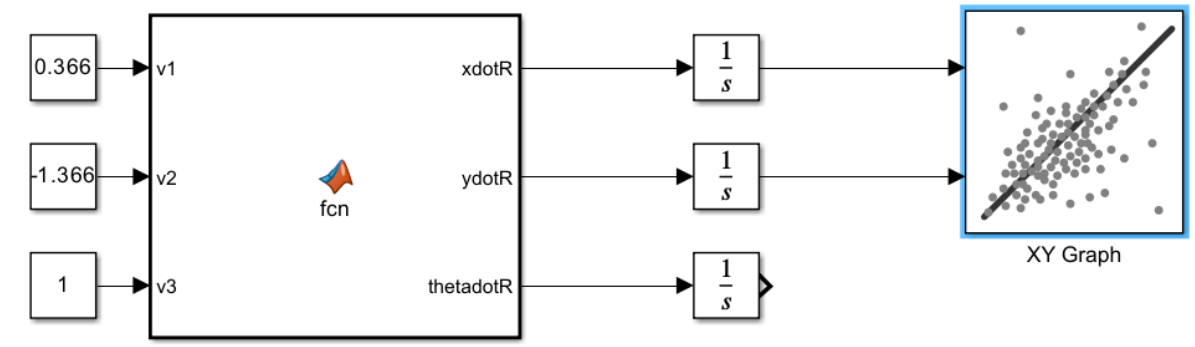


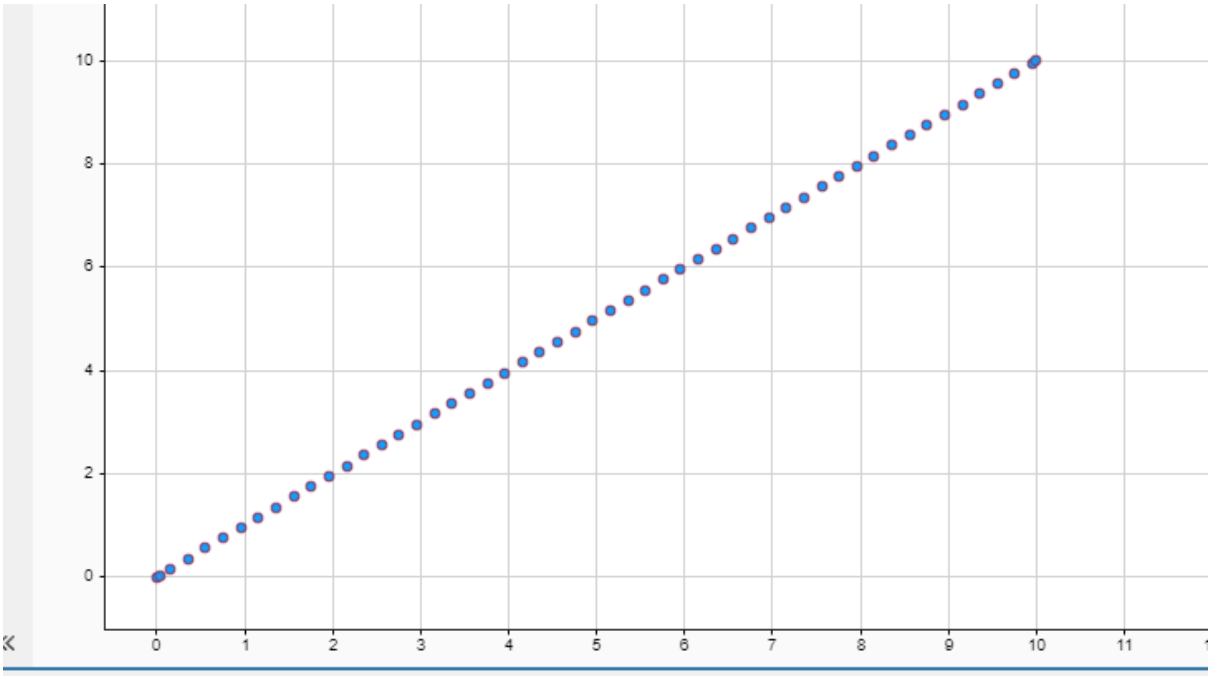
c) Calculate the three wheels' velocities so that the robot is moving in the diagonal direction in XY plane with V_{XR} = 1m/s and V_{YR} = 1m/s. The robot should not change its heading angle. Run the simulation to confirm your results

```
12    % question_c:
13    xdotR = 1;
14    ydotR = 1;
15    thetadotR = 0;
16
17    inverse_kinematics * [xdotR; ydotR; thetadotR]
```

```
Command Window

ans =

    0.3660
   -1.3660
    1.0000

fx >>
```

Verify with simulation:





II. Practice

Task 1: Write a program to control the motor to spin one round. Comment on the result (for example, how do you do that? is there any error? Why?)

We marked a point on the wheel using white tape, and then observe if the wheel rotate a whole round by observing if the mark return to its original position. According to the theory, 1 round corresponds to 700 pulses. However, this isn't true to the result we observed. We increased the number of pulses gradually, and observed that it needs 940 pulses to finish a revolution. We are unsure about why it is not 700. Maybe the sensor is just too sensitive?

```
void loop() {
  // put your main code here, to run repeatedly:
  counter = 0;
  while (counter <= 940) {
    Serial.println(counter);
    digitalWrite(motor2pin1, 128);
    digitalWrite(motor2pin2, LOW);
  }
  digitalWrite(motor2pin1, LOW);
  digitalWrite(motor2pin2, LOW);
  delay(2000);

}
```

Task 2: Write a program that controls the robot to travel a distance of 100 cm. Comment on the result (for example, how do you do that? is there any error? Why?)

Distance L to no. of revolution N_{rev} to pulses Pul:

$$N_{rev} = \frac{L}{0.1413}$$

According to measurements in task 1, $N_{rev} = 1 \Leftrightarrow Pul = 940$,
then $Pul = round(940 \times N_{rev}) = round\left(940 \times \frac{L}{0.1413}\right) = round(6652.5L)$

However, in practice, 6652 is not sufficient to drive the robot 1m. We increased up to 7000 to get the wanted result.

```
void loop() {
  resetEncoder();

  // put your main code here, to run repeatedly:
  goForward(1);
  delay(2000);

}
void goForward(float length) {
  int maxPulse = round(7000*length);
  while ((counterLA <= maxPulse) & (counterRA <= maxPulse)) {
      Serial.print("counterLA: "); Serial.print(counterLA); Serial.print("; counterRA: "); Serial.println(counterRA);
      digitalWrite(motor2pin1, LOW);
      analogWrite(motor2pin2, 255);
      digitalWrite(motor1pin1, LOW);
      analogWrite(motor1pin2, 245);
    }
  stop();
}
```

Moreover, the strength of the left and right motors are not quite balance. Specifically, the right wheel spins faster than the left wheel with the same HIGH value. Therefore, we have to approximate some value until the 2 wheels travel at the same speed, in order to get a straight line.

Task 3: Write a program to count the number of pulses generated in each second and calculate the speed of the motor (in revolutions per minute – RPM).

TASK 3_STRAIGHT LINE 1M.mp4

```
float calculateRPM(int pulseCounter) {
  float rpm = (float)pulseCounter * 60 / PULSE_PER_REV;
  return rpm;
}
```

Comment: it is important that we leave the rpm as float, not int or long, otherwise overflow will happen and shows negative value or unexpected round down of values will generate erroneous rpm value.

```
void loop() {
  unsigned long currentMillis = millis();  // Get current time
  // Check if 1 second has passed
  if (currentMillis - previousMillis >= 1000) {
    previousMillis = currentMillis;  // Save the last time pulse count was checked
    rpmLA = calculateRPM(counterLA);
    Serial.print("RPM_L = "); Serial.print(rpmLA);
    rpmRA = calculateRPM(counterRA);
    Serial.print("; RPM_R = "); Serial.println(rpmRA);
    resetEncoder();
  }
  goForward();
  Serial.print("counterLA: "); Serial.print(counterLA); Serial.print("; counterRA: "); Serial.println(counterRA);

  // delay(1000);

}
```

Task 4: Write a program to drive the robot along a circle having the diameter of 2 m.

See code attached. Demonstration here: TASK 4_FULL CIRCLE.mp4.
Initially, we attempted to find the corresponding value between 0 and 255 in order to get a ratio of 1.2 between the 2 wheels. However, it is a difficult task because the RPM result with and without the robot weight is different. In the code, we ask the robot to adjust the pwmValue (the value to analogWrite on the pin) up and down by 10 so that the rpm ratio is between the wanted value.

The drawback of this approach is that the robot will follow an unwanted path for a few seconds before it could travel in a circle. Additionally, by the way the code is written, it takes 1s between every time the pwmValue is updated according to current rpm.

```
void loop() {
  digitalWrite(motorLpin1, LOW);
  analogWrite(motorLpin2, 255);
  digitalWrite(motorRpin1, LOW);
  unsigned long currentMillis = millis();  // Get current time
  // Check if 1 second has passed
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;  // Save the last time pulse count was checked
    rpmLA = calculateRPM(counterLA);    Serial.print("RPM_L = "); Serial.print(rpmLA);
    rpmRA = calculateRPM(counterRA);    Serial.print("; RPM_R = "); Serial.println(rpmRA);
    resetCounter();
    goCircle_clockwise(1);

  }
}

void goCircle_clockwise(float radius) {
  //around a circle of 1m radius, vl=1.2vr
  float expected_speed_ratio = (radius + WHEEL_DISTANCE/2)/(radius - WHEEL_DISTANCE/2);
  Serial.print("expect ratio:");  Serial.println(expected_speed_ratio);
  float expected_rpmRA = rpmLA /expected_speed_ratio;
  Serial.print("expect rpmRA:");  Serial.println(expected_rpmRA);
  if (rpmRA < expected_rpmRA) {
      pwmValue += 10; // Increase PWM if speed is too low
      pwmValue = constrain(pwmValue, 0, 255);
    } else if (rpmRA > expected_rpmRA) {
      pwmValue -= 10; // Decrease PWM if speed is too high
      pwmValue = constrain(pwmValue, 0, 255);
    }
  analogWrite(motorRpin2, pwmValue);
  Serial.println(pwmValue);
}
```

Task 5: Write a program to drive the robot to travel half of a circle with the diameter of 2 m. Comment on the result (for example, how do you do that? is there any error? Why?)

See code attached. Demonstration here: TASK 5_HALF CIRCLE.mp4.
The approach is to calculate the distance using the wheel revolution.

```
double calculateDistance(long counterL, long counterR){
  double pulseVelocity = 0.5*(counterL+counterR);
  double distance = pulseVelocity / (double)PULSE_PER_REV * WHEEL_PERIMETER;
  return distance;
}
```

```
void loop() {
  unsigned long currentMillis = millis();  // Get current time
  // Check if 1 second has passed
  float distance = calculateDistance(totalCounterLA, totalCounterRA);
  Serial.print("distance = "); Serial.println(distance);
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;  // Save the last time pulse count was checked
    rpmLA = calculateRPM(counterLA);     Serial.print("RPM_L = "); Serial.print(rpmLA);
    rpmRA = calculateRPM(counterRA);     Serial.print("; RPM_R = "); Serial.println(rpmRA);
    resetCounter();
  }
  if (distance >= 3.14+0.3){
    stop();
    delay(3000);
    // totalCounterRA = 0;
    // totalCounterLA = 0;
  }
  goCircle_clockwise(1);
  }
```

3.14 is the length of half a circle. 0.3 is added from experience. As in task 3, even after increasing the rate from 6552 to 7000 in the code, the robot still needs to travel a few more centimeters to reach 1m. The discrepancy could be rooted in the PULSE PER REVOLUTION of 940 measured in task 1. We might have underestimated a few pulses, which accumulates to a larger error in long distance.