

RTミドルウェアによるロボットプログラミング技術

# 4. ロボットの運動学と制御の基礎 (解答)



# 演習

## ロボット制御に必要な以下のプログラムを示せ

課題 1 : 2自由度のアームの逆運動学を計算する以下の仕様の関数のC++プログラムを作成し実行結果を示せ

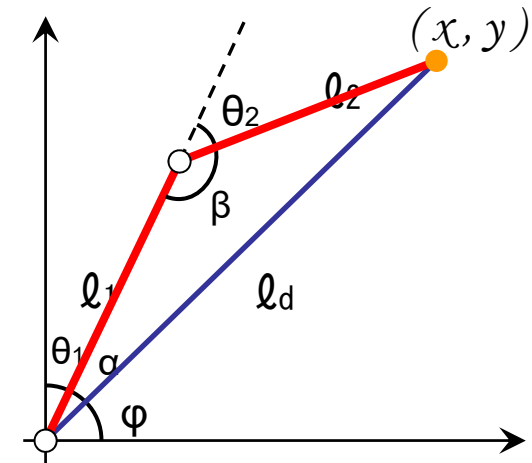
関数: `th = invkinem(link, pos)`  
th: 2つの関節の角度[deg]  
link: 2つのリンク長[m]  
pos: 手先位置[m]  
引数、戻り値はいずれも要素数2の配列とする

課題 2 : ジョイスティックの現在値から移動ロボットの車輪角速度を出力するC++プログラムを作成し実行結果を示せ

関数: `vl, vr = joy2vel (k, x, y)`  
k: 適当な係数  
x, y: ジョイスティックの現在値  
vl, vr: 移動ロボットの左右車輪角速度 [rad/s]

課題1：2自由度のアームの逆運動学を計算する以下の仕様の関数のC++プログラムを作成し実行結果を示せ

関数: Angle = invkinem(link, pos)  
 th: 2つの関節の角度[deg]  
 link: 2つのリンク長[m]  
 pos : 手先位置[m]  
 引数、戻り値はいずれも要素数2の配列とする



- 2自由度アームの制御に必要な計算式を考えてみよう

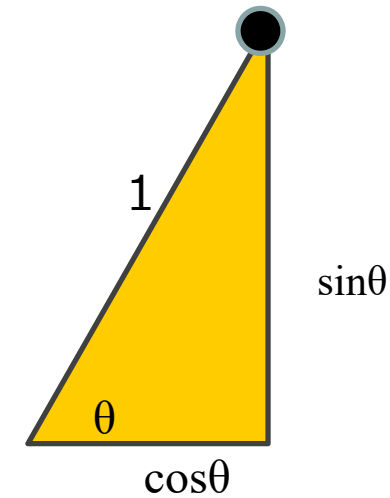
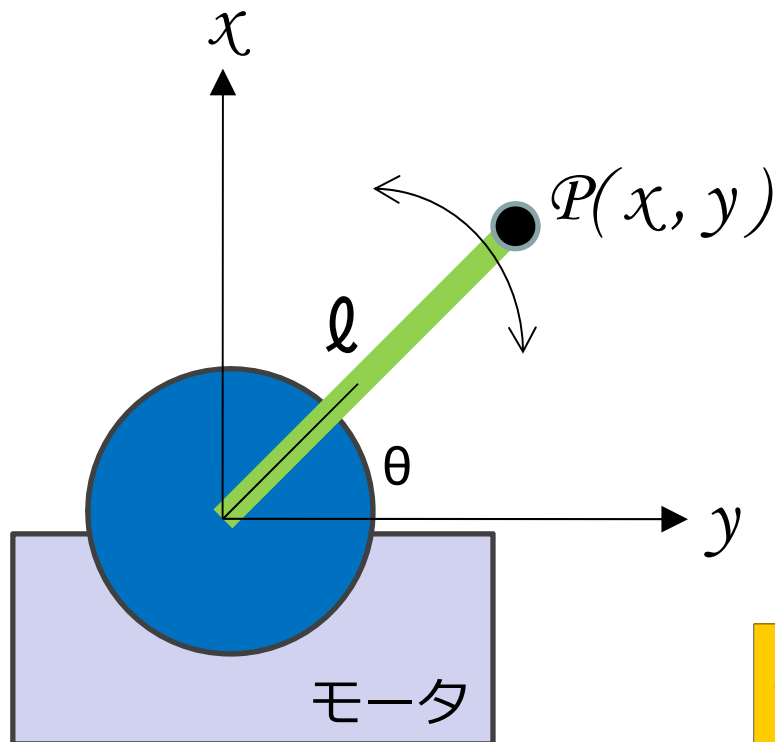
最終的には…

$\theta_1 =$

$\theta_2 =$

の形にしたい。

# 1自由度のアーム



長辺が長さ 1 の三角形の  
それぞれの辺の長さ

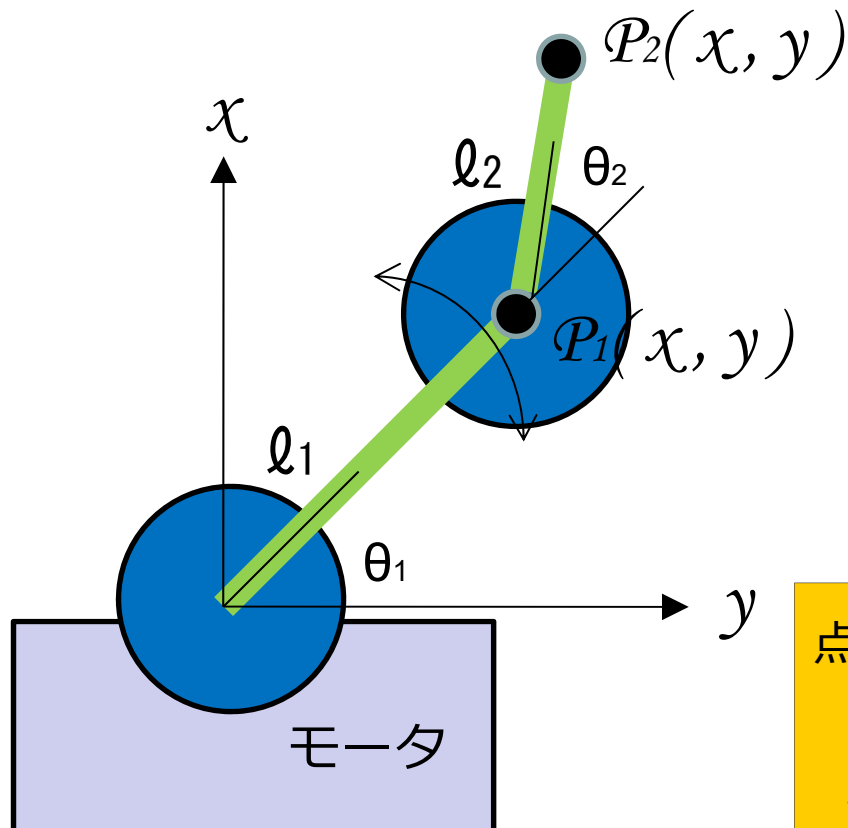
点  $P$  の位置は以下のように表せる

$$x = l \times \cos\theta$$

$$y = l \times \sin\theta$$

# 2自由度のアーム

1自由度アームの先端にもうひとつモータを繋いで2自由度アームを作る



点  $P_1$  の位置は

$$\begin{aligned} x &= l_1 \times \cos\theta_1 \\ y &= l_1 \times \sin\theta_1 \end{aligned}$$

点  $P_1$  から点  $P_2$  の位置は

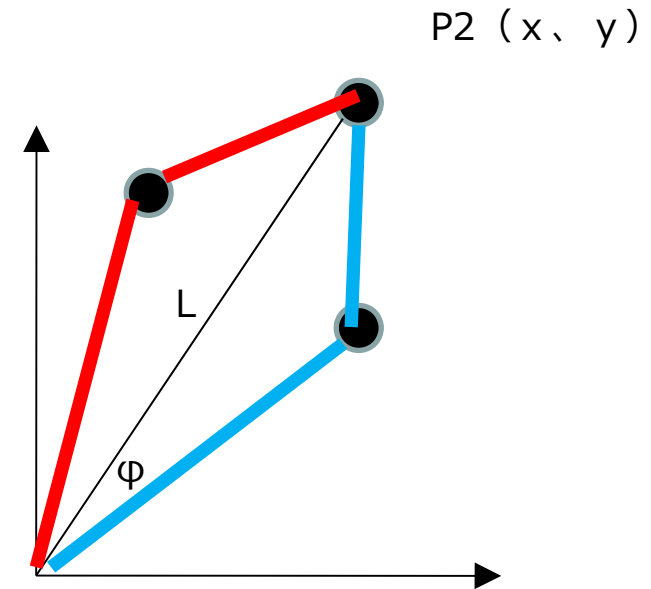
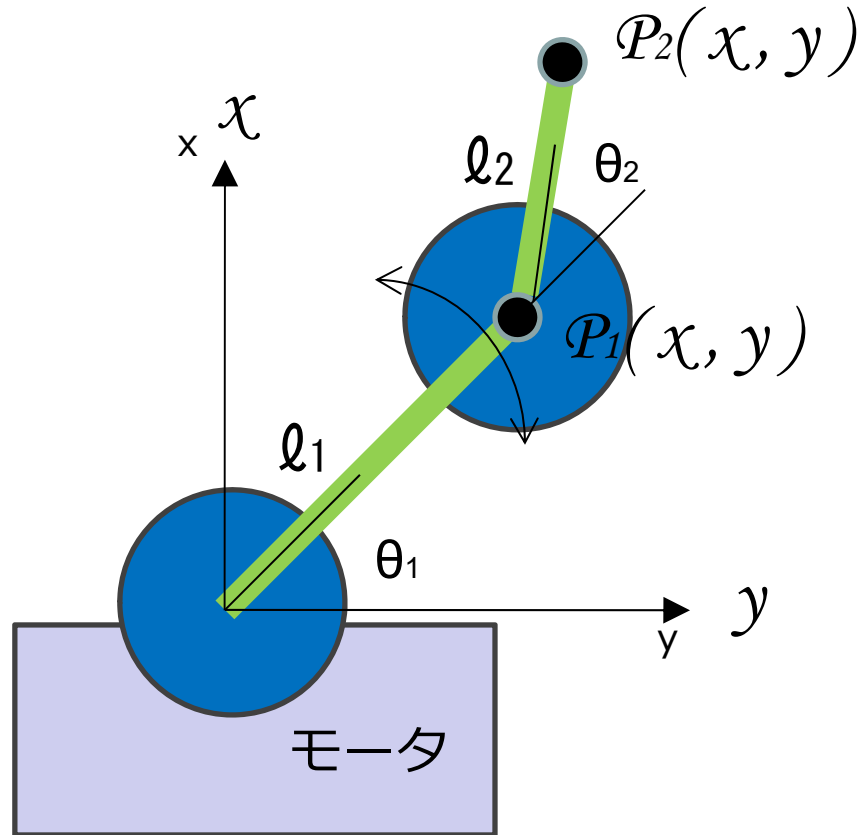
$$\begin{aligned} x &= l_2 \times \cos(\theta_1 + \theta_2) \\ y &= l_2 \times \sin(\theta_1 + \theta_2) \end{aligned}$$

点  $P_2$  の位置は以下のように表せる

$$\begin{aligned} x &= l_1 \times \cos\theta_1 + l_2 \times \cos(\theta_1 + \theta_2) \\ y &= l_1 \times \sin\theta_1 + l_2 \times \sin(\theta_1 + \theta_2) \end{aligned}$$

# 2自由度のアーム

それでは逆に点  $P_2(x, y)$  の位置にロボットの先端を持って行くにはそれぞれのモータの角度を何度にしたらよい?



ロボットアームを折り曲げて  $L$  の長さにしたものを  $\phi$  回転すると考えると2つの姿勢の可能性がある

# 解答：逆運動学

～座標から関節の角度を求める～

- 余弦定理と逆関数を使って値を求めます。

$$\theta_1 = -\left(\frac{\pi}{2} - \alpha - \varphi\right)$$

$$\tan \varphi = \frac{y}{x}$$

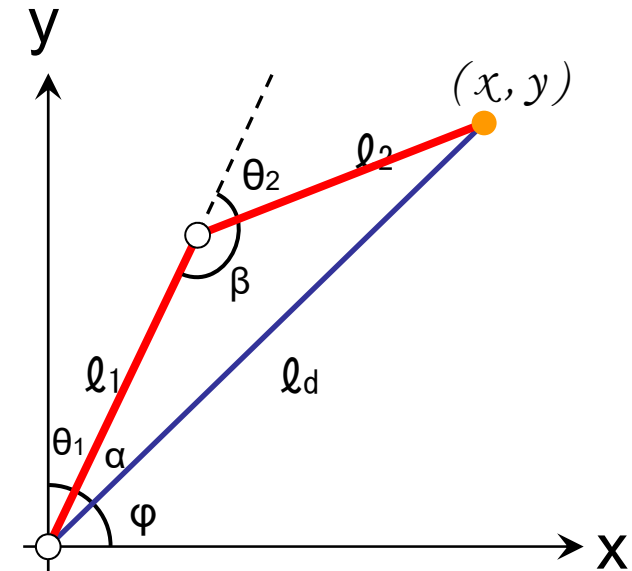
$$\theta_2 = -(\pi - \beta)$$

$$\varphi = \tan^{-1} \frac{y}{x}$$

$$\cos \alpha = \left( \frac{l_1^2 + l_d^2 - l_2^2}{2l_1 l_d} \right)$$

$$l_d = \sqrt{x^2 + y^2}$$

$$\cos \beta = \left( \frac{l_1^2 + l_2^2 - l_d^2}{2l_1 l_2} \right)$$



# 解答：逆運動学

～座標から関節の角度を求める～

- 二通りの解があります。

$$\theta_1 = -\left(\frac{\pi}{2} + \alpha - \varphi\right)$$

$$\tan \varphi = \frac{y}{x}$$

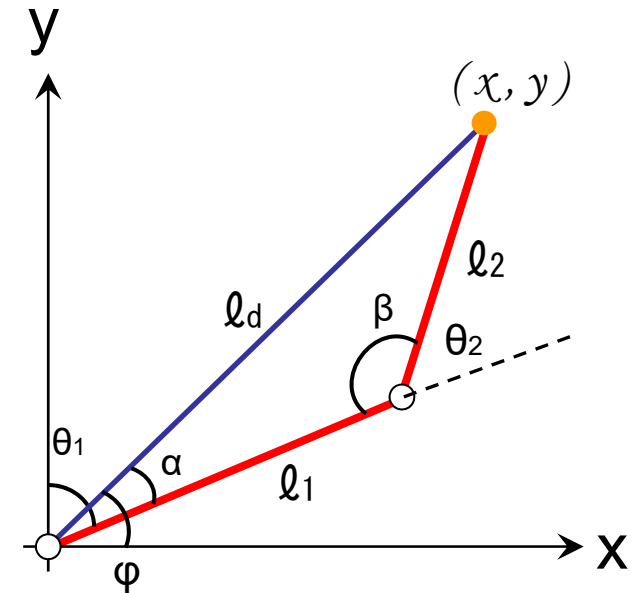
$$\theta_2 = -(-\pi + \beta)$$

$$\varphi = \tan^{-1} \frac{y}{x}$$

$$\cos \alpha = \left( \frac{l_1^2 + l_d^2 - l_2^2}{2l_1 l_d} \right)$$

$$l_d = \sqrt{x^2 + y^2}$$

$$\cos \beta = \left( \frac{l_1^2 + l_2^2 - l_d^2}{2l_1 l_2} \right)$$





# C++での実装

- #include <math.h>が必要
- $\pi$ は M\_PI (マクロ定義)
  - VC では#define \_USE\_MATH\_DEFINES が  
必要
- $\sqrt{\quad}$  は sqrt() 関数を利用
- 二乗は pow() 関数を利用
- sin, cos は sin(), cos() 関数を利用
- $\tan^{-1}$ は atan2() 関数を利用
- $\cos^{-1}$ は acos() 関数を利用

# プログラム (C++) (1)

```

JointAngle invkinem(LinkLength link, Position pos)
{
    double ld(sqrt((pos.x * pos.x) + (pos.y * pos.y)));
    double l1(link.l0);
    double l2(link.l1);

    double a = acos((pow(l1, 2) + pow(ld, 2) - pow(l2, 2)) / (2 * l1 * ld));
    double b = acos((pow(l1, 2) + pow(l2, 2) - pow(ld, 2)) / (2 * l1 * l2));
    double phi = atan2(pos.y, pos.x);

    JointAngle j;
    j.th0 = - ((M_PI / 2) - a - phi) * 180 / M_PI;
    j.th1 = - ( M_PI - b) * 180 / M_PI;

    return j;
}

```

# プログラム (C++) (2)

```

JointAngle invkinem2(LinkLength link, Position pos)
{
    double ld(sqrt((pos.x * pos.x) + (pos.y * pos.y)));
    double l1(link.l0);
    double l2(link.l1);

    double a = acos((pow(l1, 2) + pow(ld, 2) - pow(l2, 2)) / (2 * l1 * ld));
    double b = acos((pow(l1, 2) + pow(l2, 2) - pow(ld, 2)) / (2 * l1 * l2));
    double phi = atan2(pos.y, pos.x);

    JointAngle j;
    j.th0 = - ((M_PI / 2) + a - phi) * 180 / M_PI;
    j.th1 = - (-M_PI + b) * 180 / M_PI;

    return j;
}

```

# 参考：順運動学（2自由度）

点P<sub>1</sub>の位置は、

$$x_1 = l_1 \cos(-\theta_1)$$

$$y_1 = l_1 \sin(-\theta_2)$$

点P<sub>1</sub>から点P<sub>2</sub>の位置は、

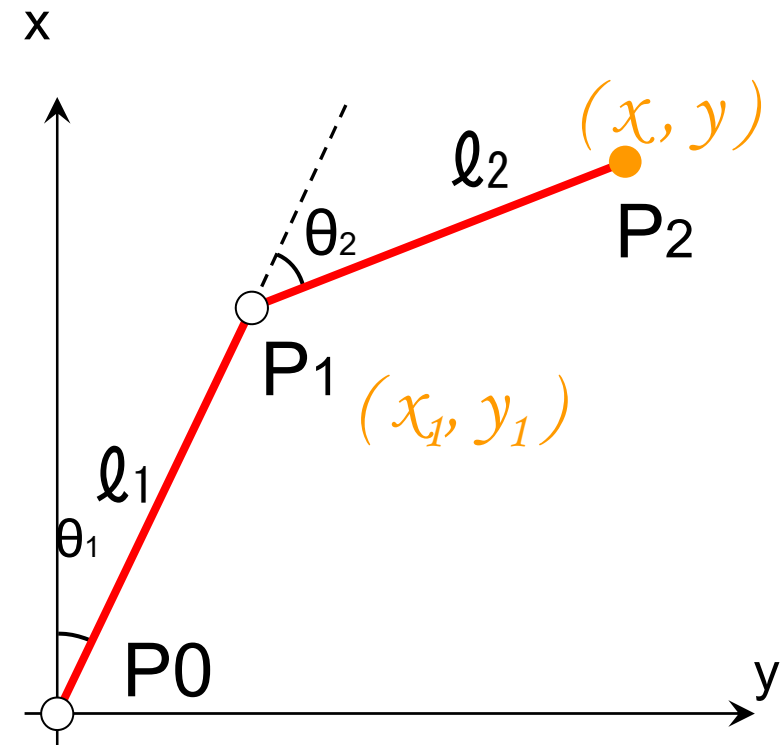
$$x_2 = l_2 \cos(-\theta_1 - \theta_2)$$

$$y_2 = l_2 \sin(-\theta_1 - \theta_2)$$

もとめる手先の位置は、

$$x = x_1 + x_2$$

$$y = y_1 + y_2$$



# 解答

```

$ ./arm2dof
pos (x, y): -1, 1      ==> angle (th0, th1): 90, -90
pos (x, y): -0.5, 1   ==> angle (th0, th1): 82.5772, -112.024
pos (x, y): 0, 1      ==> angle (th0, th1): 60, -120
pos (x, y): 0.5, 1    ==> angle (th0, th1): 29.4471, -112.024
pos (x, y): 1, 1      ==> angle (th0, th1): -1.27222e-14, -90
または
$ ./arm2dof
pos (x, y): -1, 1      ==> angle (th0, th1): -0, 90
pos (x, y): -0.5, 1   ==> angle (th0, th1): -29.4471, 112.024
pos (x, y): 0, 1      ==> angle (th0, th1): -60, 120
pos (x, y): 0.5, 1    ==> angle (th0, th1): -82.5772, 112.024
pos (x, y): 1, 1      ==> angle (th0, th1): -90, 90

```

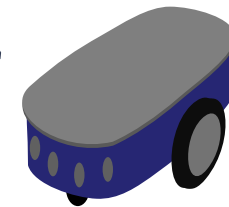
# 課題 2 : ジョイスティックの現在値から移動ロボットの車輪角速度を出力するPythonプログラムを作成し実行結果を示せ

関数:  $v_l, v_r = \text{joy2vel}(k, x, y)$

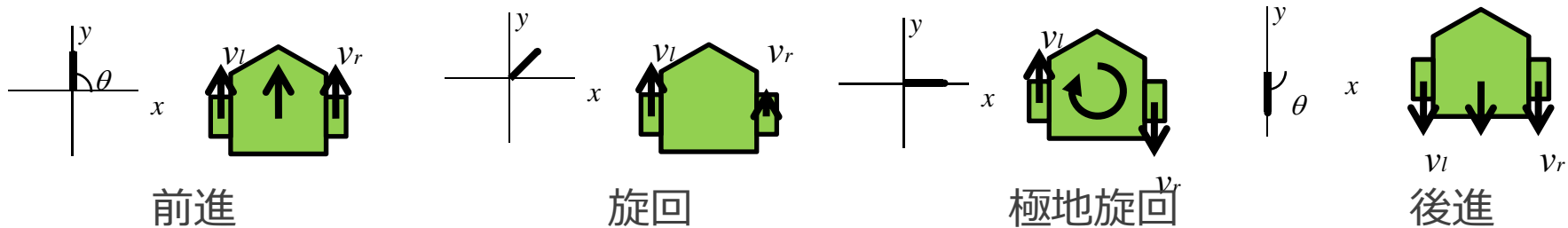
$k$ : 適当な係数

$x, y$ : ジョイスティックの現在値

$v_l, v_r$ : 移動ロボットの左右車輪角速度 [rad/s]

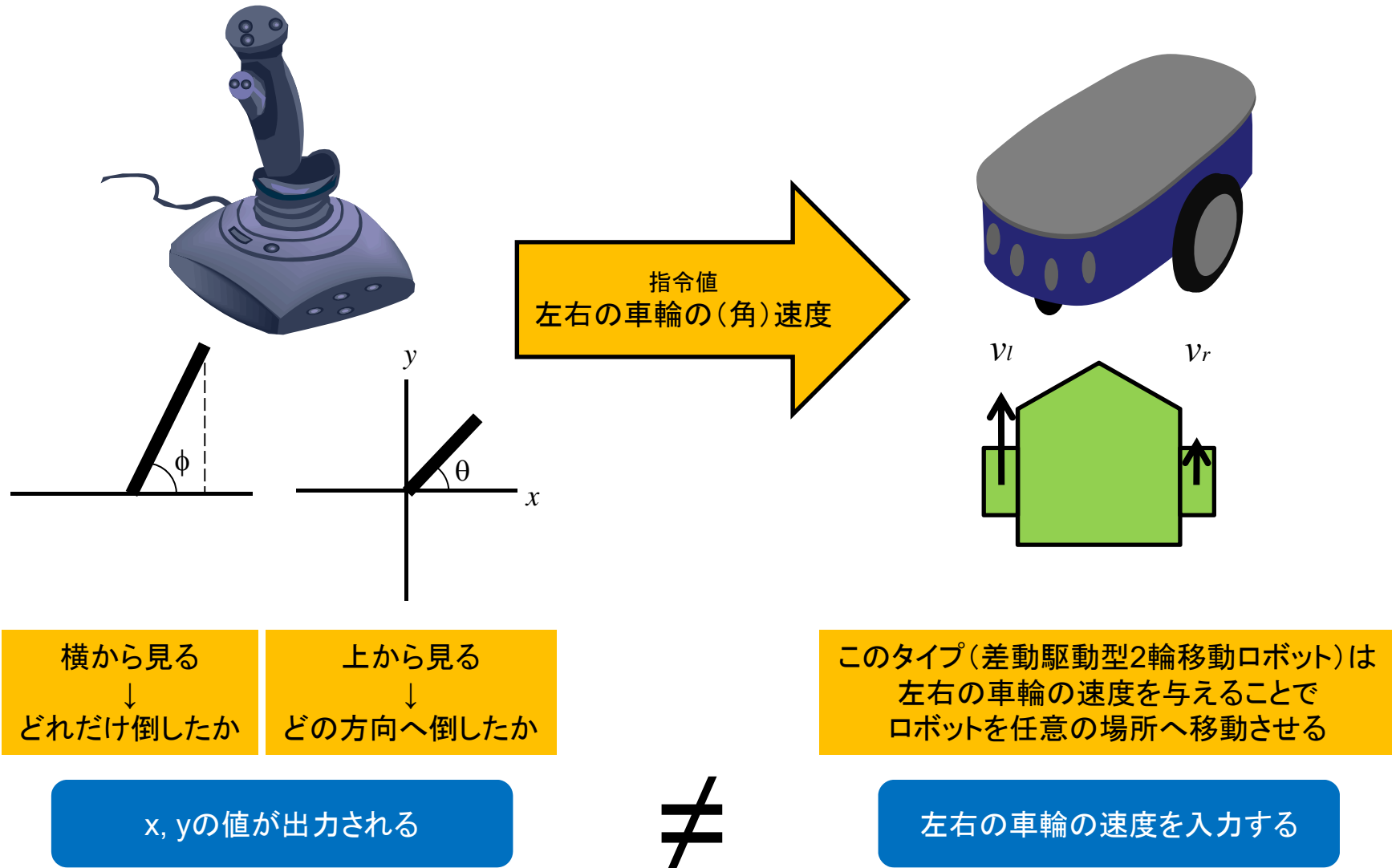


ジョイスティックを倒す角度と車輪の速度の大まかな関係は…

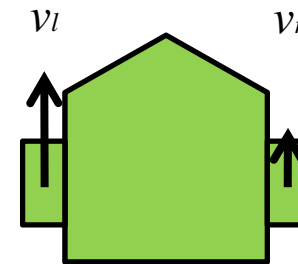
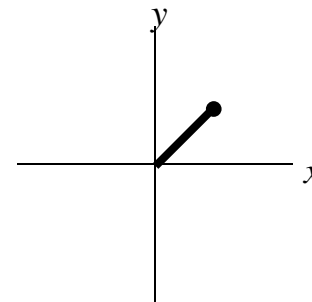
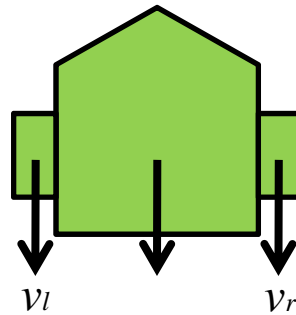
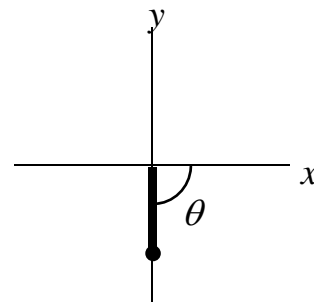
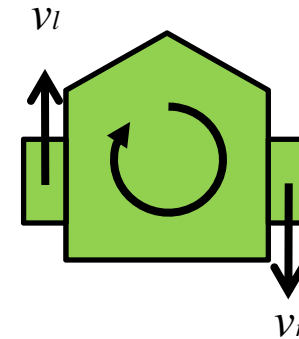
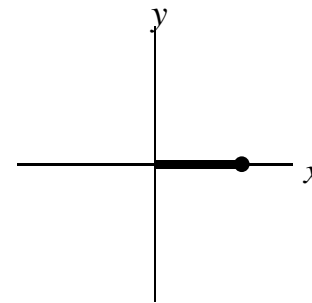
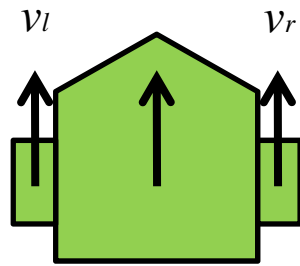
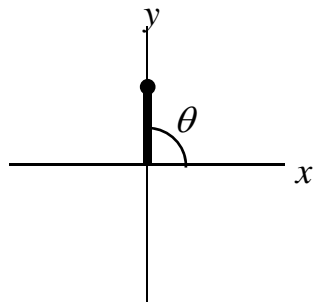


これらを滑らかにつなぐ関数を定義したい。

# ジョイスティックによる操作

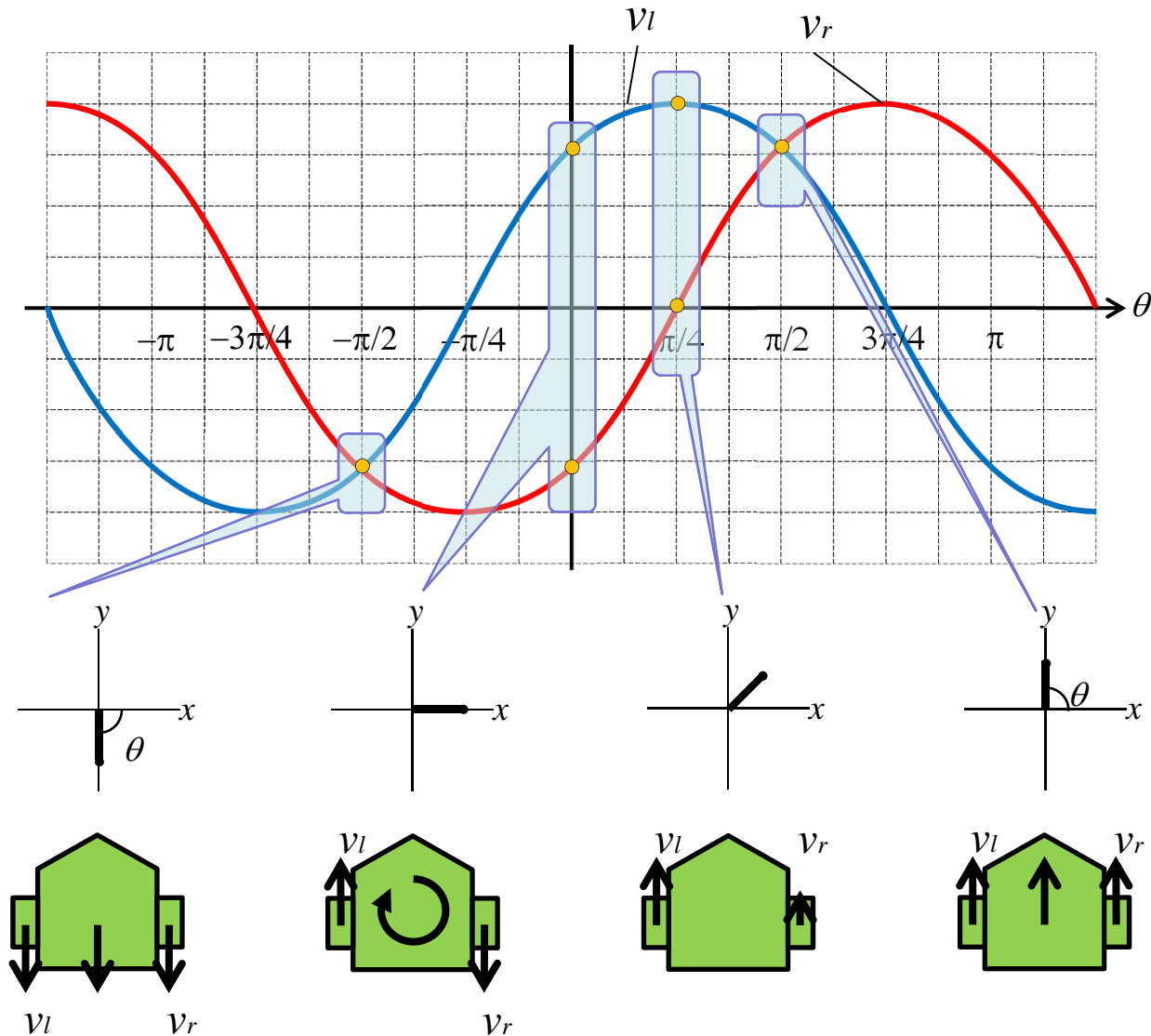


# 移動ロボットの制御

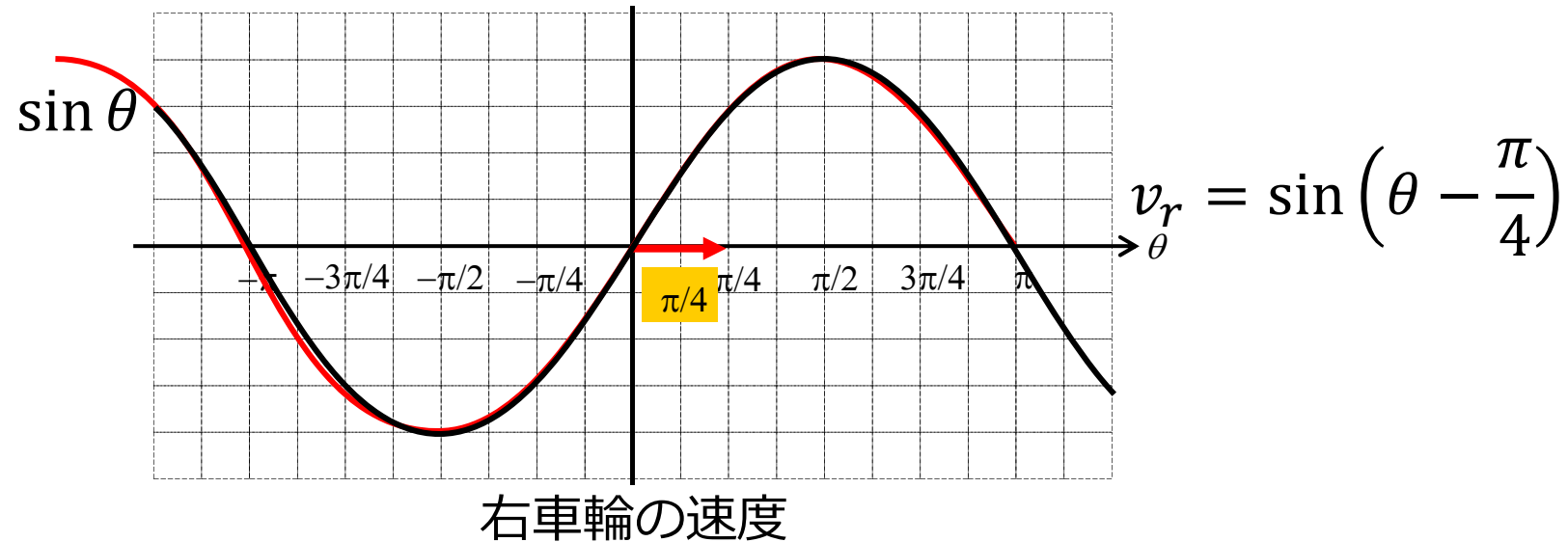
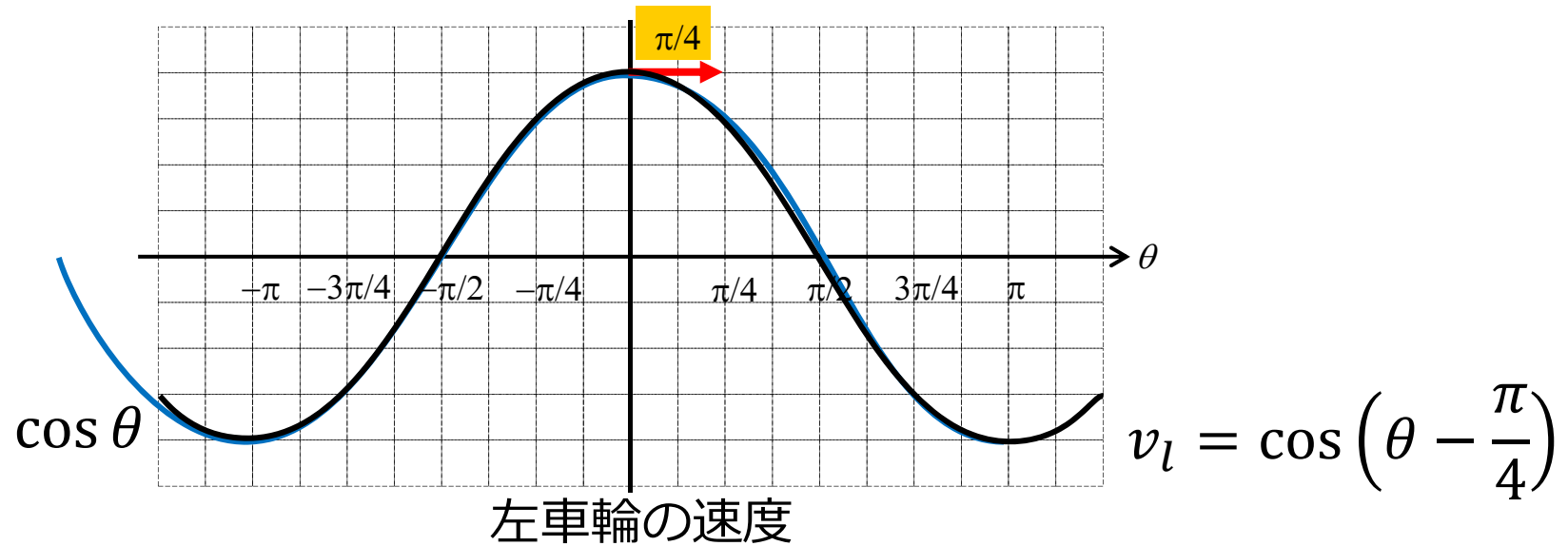




# ジョイスティックと車輪の速度



# 解答



# プログラム (C++)

```
WheelVelocity joystick(double k, Position pos)
{
    double th = atan2(pos.y, pos.x);
    double v = k * hypot(pos.x, pos.y);
    WheelVelocity wv;
    wv.v_l = v * cos(th - M_PI / 4.0);
    wv.v_r = v * sin(th - M_PI / 4.0);

    return wv;
}
```

# 解答

```
$ ./joystick  
pos (x, y): 0, 1      ==> velocity (v_l, v_r): 0.707107, 0.707107  
pos (x, y): 1, 1      ==> velocity (v_l, v_r): 1.41421, 0  
pos (x, y): 1, 0      ==> velocity (v_l, v_r): 0.707107, -0.707107  
pos (x, y): 0, -1     ==> velocity (v_l, v_r): -0.707107, -0.707107  
pos (x, y): -1, -1    ==> velocity (v_l, v_r): -1.41421, -1.73191e-16
```