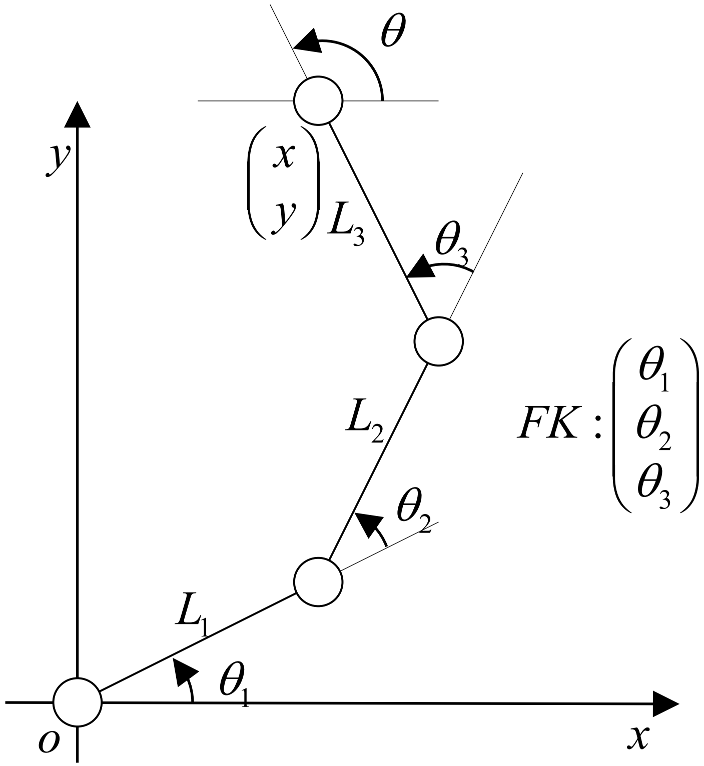
ロボットアームの順運動学: C++によるコーディング
Robot Arm Forward Kinematics: C++ Coding

成瀬継太郎（会津大）
Keitaro Naruse (Univ. of Aizu)



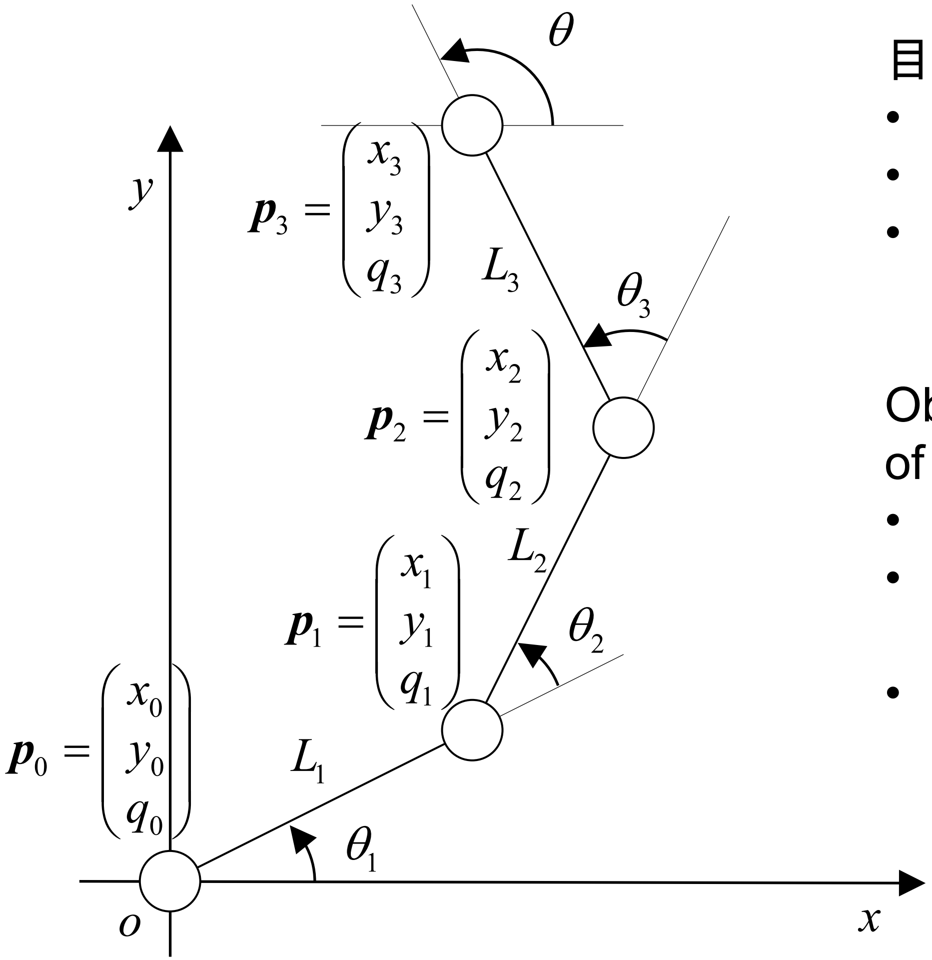
$$FK : \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix} \to \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

まとめ
- EigenとC++による順運動学関数のコーディング

Summary
- C++ coding of a forward kinematics function with Eigen

ロボットアームの順運動学関数のC++コーディング

C++ Coding of Forward Kinematics Function of Robot Arm

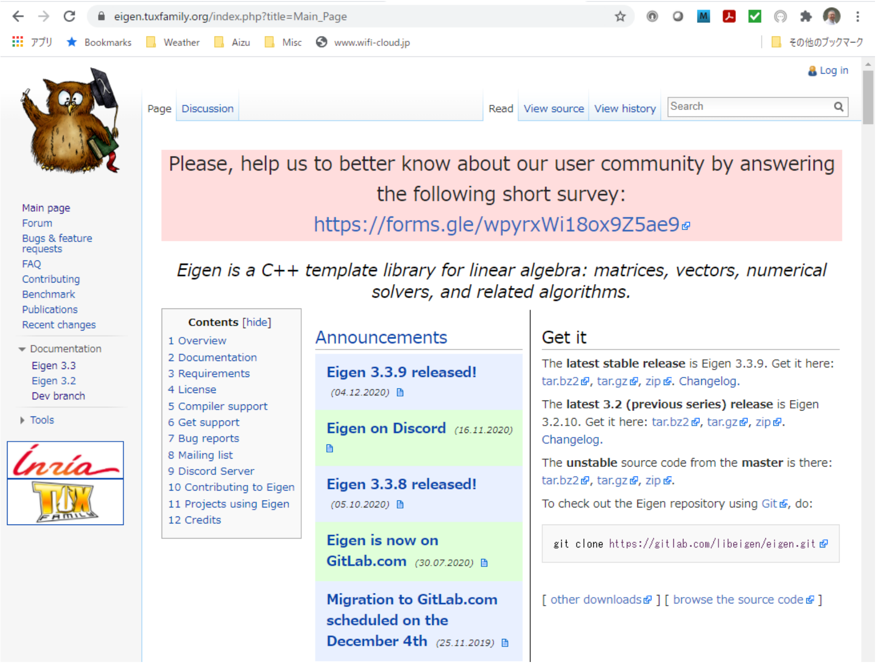


目的:ロボットアームの順運動学をC++の関数として実装する
- 引数:関節角度ベクトル
- 戻り値:姿勢行列(各列が各関節の姿勢を表す)
- 行列とベクトルを使ったコードになるため，外部ライブラリとしてEigenを導入する

Objective: To implement a C++ function to solve forward kinematics of a robot arm
- Argument: A joint angle vector
- Return: A pose matrix, in which a column represents a pose of a joint
- We introduce an external library called Eigen, which can handle the linear algebra such as matrices and vectors

$$\boldsymbol{q} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix} \longrightarrow \boldsymbol{p} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3) = \left(\begin{array}{c:c:c:c} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ q_0 & q_1 & q & q_3 \end{array}\right)$$

Eigenの導入
Introduction of Eigen



Eigen site: https://eigen.tuxfamily.org/

Eigenは線形代数のためのC++のテンプレートライブラリ．行列，ベクトル，数値解法，関連アルゴリズムが利用可能
- 簡単な導入：ヘッダファイルをインクルードするだけで，外部ライブラリをリンクする必要がない

Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms.
- Easy to introduce: Just include header files, and no need to specify external libraries, because it is a template library.

Linuxへのインストール方法 / Linux install
- sudo apt install libeigen3-dev
- Eigen is installed at /usr/include/eigen3/
- Just g++ as you do in a regular C++ source code

ウィンドウズユーザへ，WSL/WSL2をインストールして，Ubuntu18.04上で開発することをお勧めします
Windows users: I recommend you install WSL/WSL2 and develop it in Linux

必要なインクルードファイルと順運動学関数
Required Include Files and Forward Kinematics Function

```
#include <iostream>
#include <cmath>
#include <eigen3/Eigen/Dense>


Eigen::Matrix<double, 3, 4> fk(const Eigen::Vector3d& q)
{
    //  Robot arm link parameters
    const double L1 = 1.0, L2 = 1.0, L3 = 1.0;

    //  A pose vector of joints and hand
    Eigen::Vector3d p0, p1, p2, p3;

    //  A vector of the pose vectors as a matrix
    //  p = (p0, p1, p2, p3)
    Eigen::Matrix<double, 3, 4> p;
```

コンソール出力のため
For console out

三角関数のため
For cos() and sin() out

Eigenによるベクトルと行列表現のため
For vectors and matrices with Eigen

戻り値:ロボットアームの姿勢 double型3*4行列 = 各姿勢が3要素(x,y,q) * 4姿勢
Return: A robot arm pose of a double 3*4 matrix = each pose has 3 components of (x,y,q) * 4 poses

$$\boldsymbol{p} = \left(\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3\right) = \left(\begin{array}{c:c:c:c} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ q_0 & q_1 & q_2 & q_3 \end{array}\right)$$

引数:関節角度ベクトル double型3要素ベクトル
Argument: A joint angle vector of 3 double components

$$\boldsymbol{q} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix}$$

Code is available at https://github.com/keitaronaruse/Naruse-robotics-tutorial/blob/main/src/cpp/fk-3link-planar.cc

順運動学関数の続き
Forward Kinematics Function (Continued)

```
    //  Forward kinematics calculation
    //  p0: A pose of the first joint = the base
    p0 << 0.0, 0.0, 0.0;
    p.col(0) = p0;

    //  p1: A pose of the second joint = the end of the first link
    p1 << L1 * std::cos(q(0)), L1 * std::sin(q(0)), q(0); p1 = p0 + p1;
    p.col(1) = p1;

    //  p2: A pose of the third joint = the end of the second link
    p2 << L2 * std::cos(q(0)+q(1)), L2 * std::sin(q(0)+q(1)), q(1); p2 = p1 + p2;
    p.col(2) = p2;

    //  p3: A pose of the hand tip = the end of the third link
    p3 << L3 * std::cos(q(0)+q(1)+q(2)), L3 * std::sin(q(0)+q(1)+q(2)), q(2); p3 = p2 + p3;
    //  Assign p3 to the 3rd column of the matrix p
    p.col(3) = p3;

    return(p);
}
```

$$\boldsymbol{p} = \left(\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3\right) = \left(\begin{array}{c:c:c:c} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ q_0 & q_1 & q_2 & q_3 \end{array}\right)$$

順運動学関数の実行
Run Forward Kinematics Function

```
int main()
{
    //  A joint angle vector (q1, q2, q3)
    Eigen::Vector3d q;
    //  Initial value is (0.1, 0.4, 0.9) [rad]
    q << 0.1, 0.4, 0.9;

    //  A set of poses as a matrix
    //  p = (p0, p1, p2, p3)
    Eigen::Matrix<double, 3, 4> p;

    //  Forward kinematics solution
    p = fk(q);

    //  Console out the joint angle vector
    std::cout << q << std::endl;

    //  Console out the pose matrix = a vector of poses
    std::cout << p << std::endl;
    return(0);
}
```