Robot Arm Model
（ロボットアームモデル）

$$p = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\theta = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$$

θ_2

θ_1
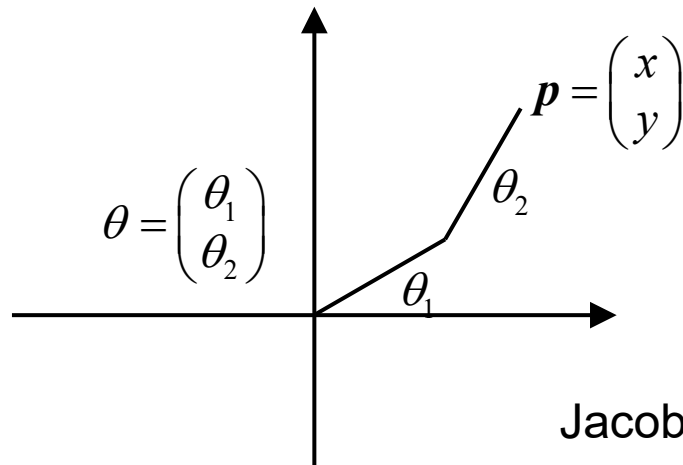
Forward kinematics

$$x = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2)$$
$$y = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2)$$
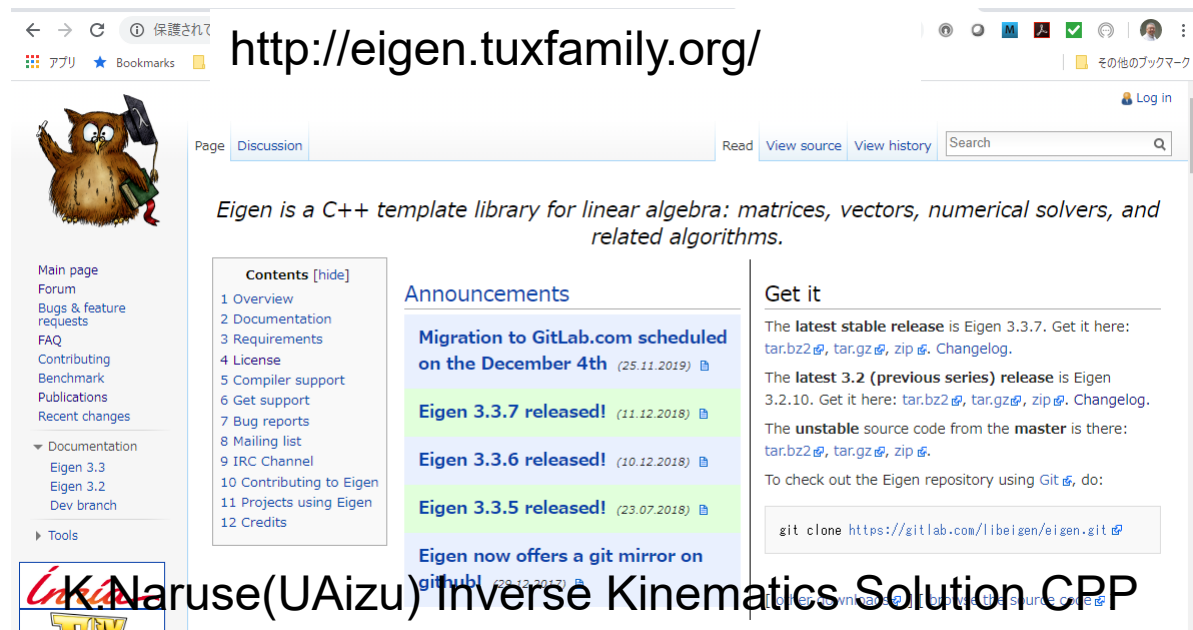
Jacobian and inverse of Jacobian

$$\dot{p} = J\dot{\theta}$$

$$J = \begin{pmatrix} j_{11} & j_{12} \\ j_{22} & j_{22} \end{pmatrix} = \begin{pmatrix} -L_1 \sin(\theta_1) - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{pmatrix}$$

$$J^{-1} = \frac{1}{j_{11}j_{22} - j_{12}j_{22}} \begin{pmatrix} j_{22} & -j_{12} \\ -j_{22} & j_{11} \end{pmatrix}$$

K.Naruse(UAizu) Inverse Kinematics Solution CPP Source Code

Sample Codes for Numerical Calculation
（数値計算に関するソースコード）

Filename	Description
ik_2link_pure_cpp.cpp	Linear algebra is implemented with C++ （線形代数もC++のみで書かれている）
ik_2link_symJ_eigen.cpp	Linear algebra is implemented with Eigen, a c++ math library （線形代数はEigenで書かれている）

http://eigen.tuxfamily.org/

K.Naruse(UAizu) Inverse Kinematics Solution CPP Source Code

Sample Codes for Choreonoid
（コレオノイド用のコード）

Filename	Description
xy2link.body	Body file of 2-link arm for Choreonoid （2リンクロボットアームのボディファイル）
XY2link.Controller1.cpp	Simple controller of controlling a joint angle to (30, 30)[deg] （関節角度制御だけを行う）
XY2link.Controller2.cpp	Simple controller of controlling a joint angle which satisfies a hand position of (0.0, 0.1) （手先の位置に対する関節角度は別で計算され，ここでは関節角度制御だけを行う）
XY2link.Controller3.cpp	Simple controller of solving inverse kinematics in initialization() and controlling a joint angle to the solved one in control() （逆運動学はinitialization()で関節角度制御はcontrol()で実行する）
XY2link.Controller4.cpp	Simple controller of solving inverse kinematics and controlling a joint angle both in control() （逆運動学と関節角度制御の両者をcontrol() で実行する）

Let Look at Codes
（それでは，コードを見てみよう）

K.Naruse(UAizu) Inverse Kinematics Solution CPP Source Code