Interdisciplinary Research Fundamentals II of STM

Class 3

人間医療科学技術他分野専門基礎第二

クラス3

Day 3

2026-06-25

Takahiro Kanno, RIVERFIELD Inc.

菅野 貴皓（リバーフィールド株式会社）

Declaration of AI-Generated Content

In this presentation material, AI has been used for the following purposes:
- Generating some images such as illustrations and backgrounds
- Translation
- Content review

This icon indicates AI-generated image

This icon is also generated by AI

Last week...

We covered the electric parts in robots. We focused on:

- Overview of electric parts
- Principle of various motors
- Position sensors
- Force sensors and laser obstacle sensors
- Electrical communication inside robots

先週：ロボットの電気要素（センサとアクチュエータ）

- 電気要素の概要
- 各種モーターの動作原理
- 位置センサ
- 力センサとレーザー式障害物センサ
- ロボット内部の電気通信

Today: algebra and kinematics

Linear algebra is essential for controlling a robot.

Robot control technology includes kinematics, classical control theory, and modern control theory

Other necessary mathematics:

- Trajectory optimization
- Image processing
- Machine learning

今日の内容：代数学と運動学

線形代数はロボット制御に不可欠です。

トピック：運動学、古典制御理論、現代制御理論

ロボット工学に必要なその他の数学：

- 軌道最適化
- 画像処理
- 機械学習

What is “Kinematics”?

Kinematics

- “Kinematics is a subfield of classical mechanics that describes the motion of points, bodies (objects), and systems of bodies (groups of objects) without considering the forces that cause them to move.” (Wikipedia)
- In robotics, kinematics often refers to describing the relationship between a robot’s positions and postures and their corresponding actuator displacements.

Dynamics

- Based on equations of motion
- Considers inertia and acceleration

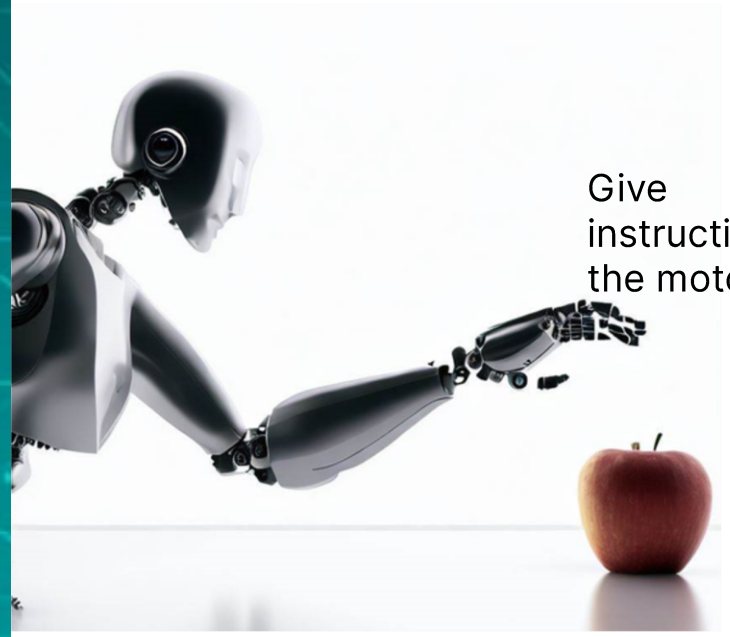
運動学

- 「運動学は、力がそれらを動かす原因を考慮せずに、点・物体、または物体系（複数の物体）の運動を記述する古典力学の一分野である。」
- ロボット工学では、運動学はロボットの位置や姿勢と、それに対応するアクチュエータの変位との関係を表すことを指すことが多い。

動力学

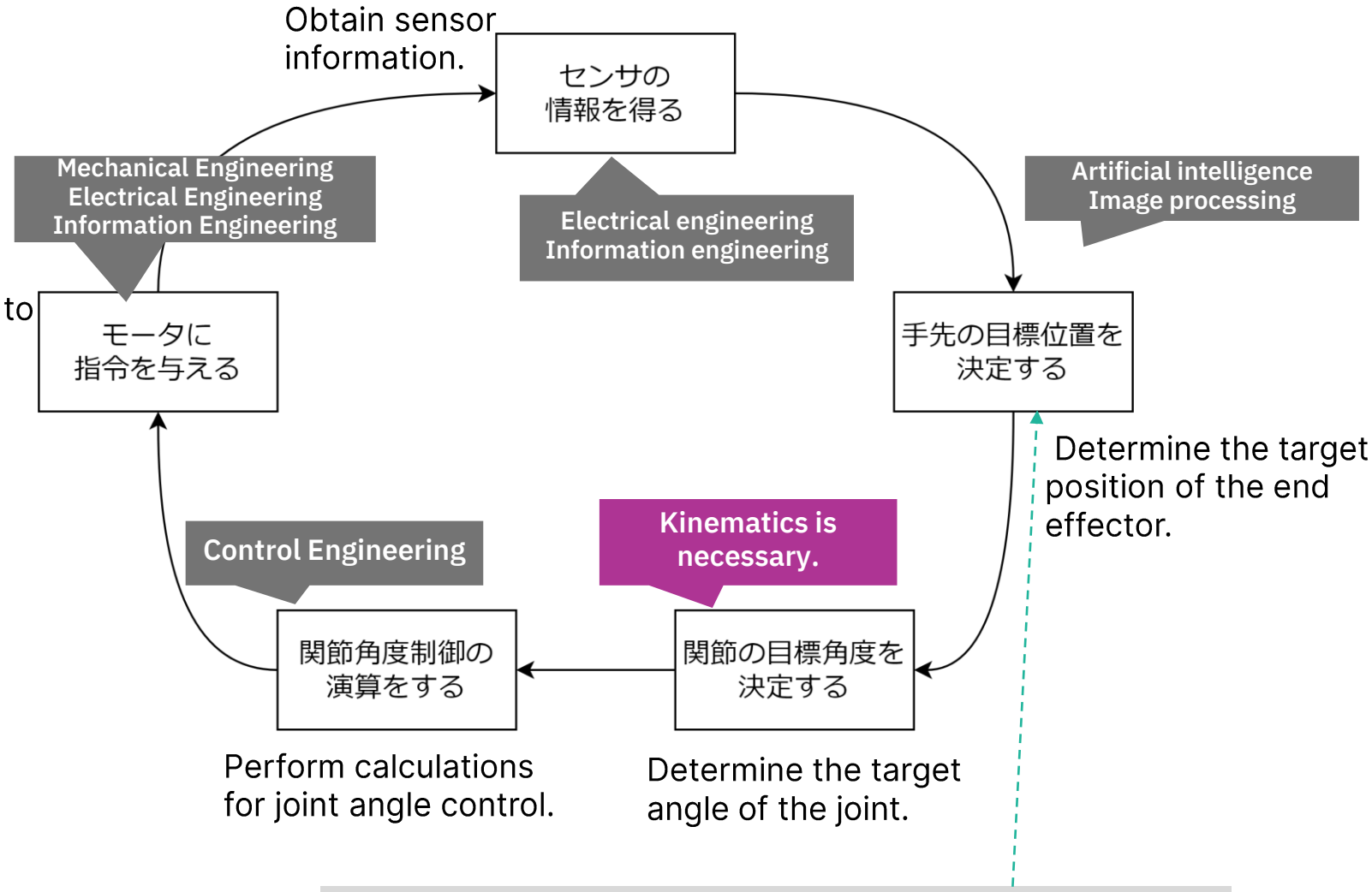
- 運動方程式に基づく
- 慣性と加速度を考慮する

The overall view of robot control



AI-generated image

Give instructions to the motor.



- Determined by image processing and AI
- Pre-programmed in advance
- User gives instructions in real-time via remote control

Forward kinematics and inverse kinematics

Forward kinematics

Mathematical methods for determining hand position from joint angles.

Inverse kinematics

Mathematical methods for determining joint angles from hand position.

- Calculate the user's hand position from the angle data of a 3D input device

Geomagic Touch

- Convert a robot's end-tool position command into joint-angle commands
- In computer graphics or game development, calculate arm and leg angles from a character's hand and foot positions.

順運動学 (Forward Kinematics)
順運動学は、関節角度から手の位置を求める数学的手法です。
例：3D入力装置（Geomagic Touchなど）の角度データからユーザーの手の位置を計算する。

逆運動学 (Inverse Kinematics)
逆運動学は、手の位置から関節角度を求める数学的手法です。
例：ロボットの末端工具（エンドツール）の位置指令を関節角度の指令に変換する。
例：CG（コンピュータグラフィックス）では、キャラクターの手や足の位置から腕や脚の角度を計算する。

One-link kinematics example 1リンクの運動学

- **Problem:** Given the joint angle θ, find the tip position (x, y).
  関節角度 θ が与えられているとき, 先端位置 (x, y) を求める

Forward Kinematics

- $x = L\cos\theta$
- $y = L\sin\theta$

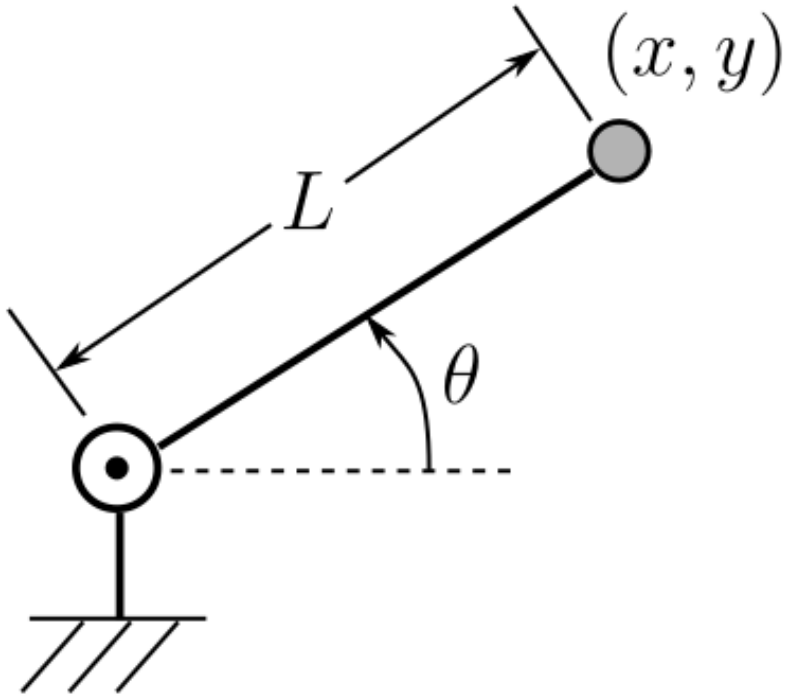
Inverse Kinematics

- Solve the above equations for θ

$$\frac{y}{x} = \frac{L\sin\theta}{L\cos\theta} = \tan\theta$$

↓

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



8

Two-link kinematics example

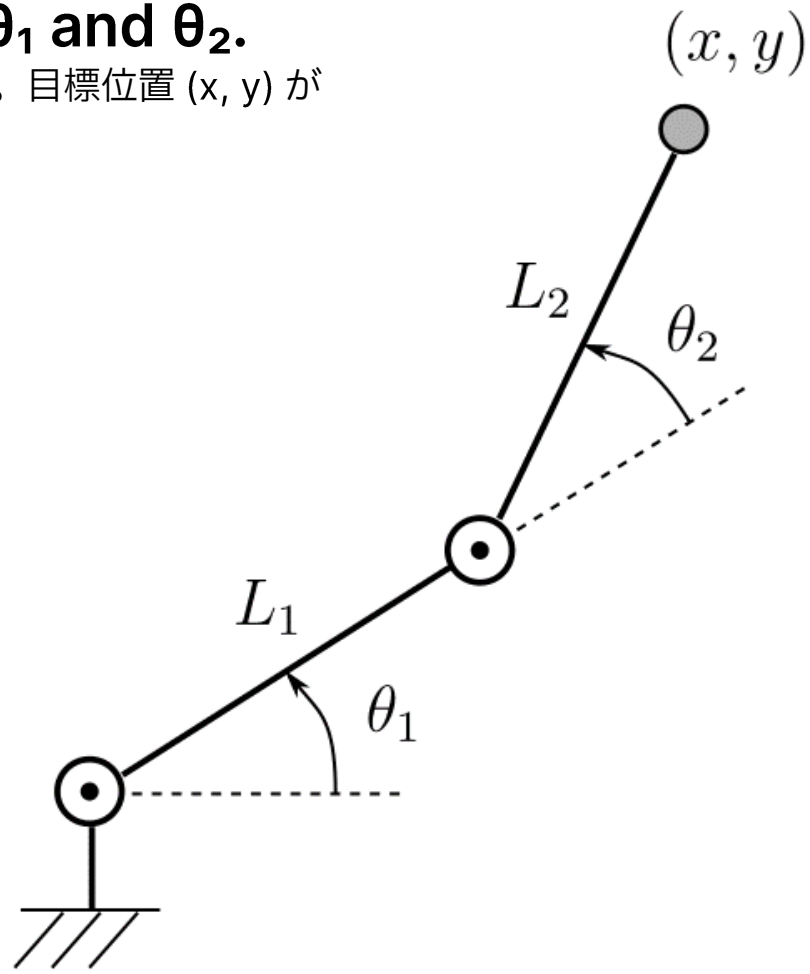
A two-link robot has two straight arms of lengths L_1 and L_2 joined in series. Given a target point (x, y), we find joint angles θ_1 and θ_2.

2リンクロボットは長さ L_1 と L_2 の2本の直線アームが直列に連結されています。目標位置 (x, y) がわかっているとき、関節角 θ_1 と θ_2 を求めます。

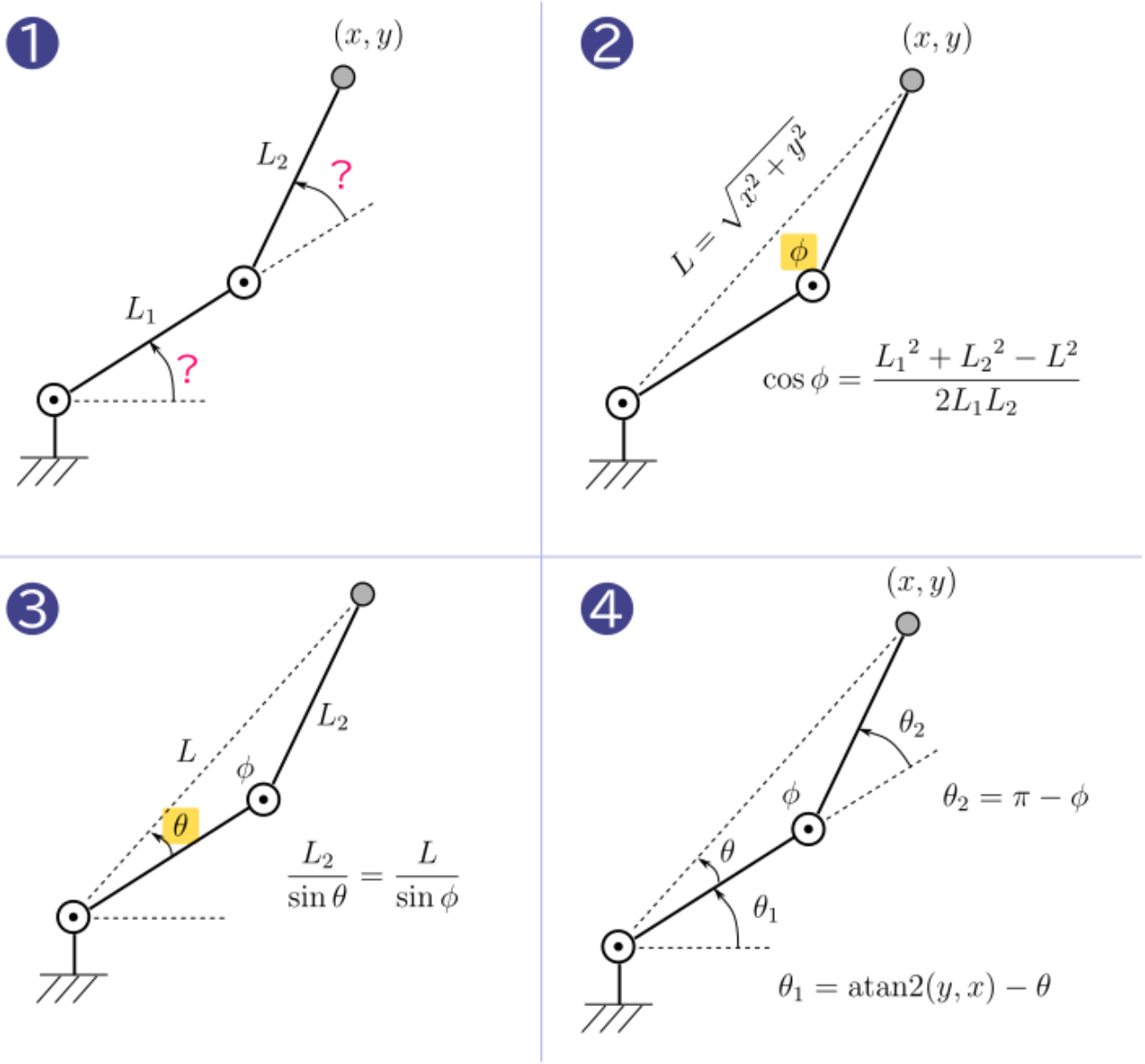
Forward kinematics

- $x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$
- $y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$

Inverse kinematics ???



Two-link inverse kinematics



① Setup

Draw the links and the point (x, y).

Mark the two unknown angles (at the base and at the elbow).

アーム（長さ L_1, L_2）と目標点 (x, y) を描く。

ベースと肘の関節角 θ_1, θ_2 を求める。

② Cosine theorem

Compute the straight-line distance r from the base to (x, y):

ベースから (x, y) までの距離 r を計算

$$r = \sqrt{x^2 + y^2}.$$

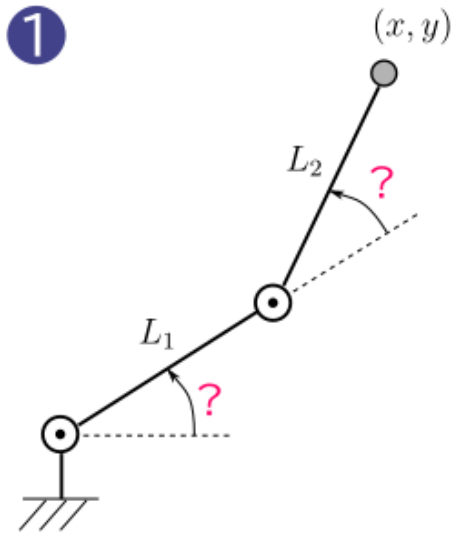
In the triangle of sides L_1, L_2, and r, use

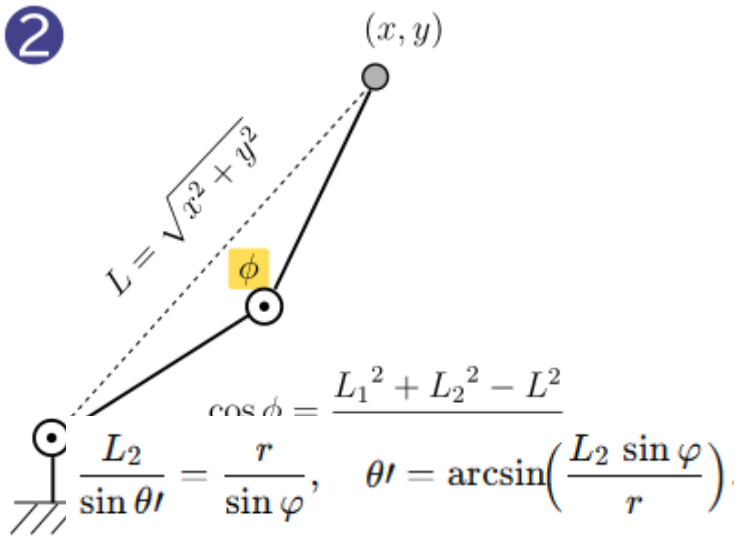
L_1, L_2, r の三角形で余弦定理を使う

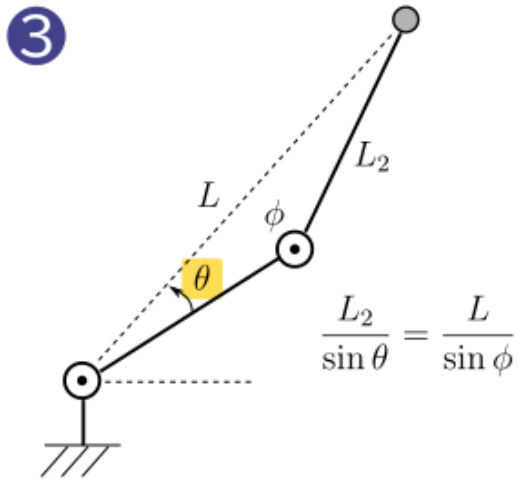
$$\cos\varphi = \frac{L_1^2 + L_2^2 - r^2}{2\,L_1\,L_2}, \quad \varphi = \arccos\left(\frac{L_1^2 + L_2^2 - r^2}{2\,L_1\,L_2}\right)$$

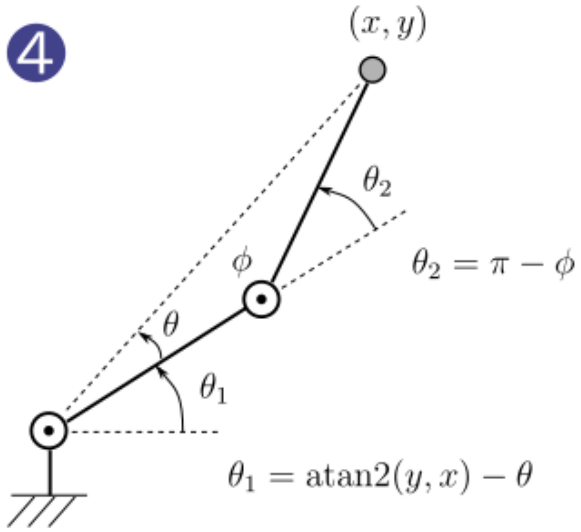
10

Two-link inverse kinematics









③ Sine theorem

Still in the same triangle, relate side L_2 and angle φ to r:

同じ三角形で正弦定理を適用

$$\frac{L_2}{\sin\theta\prime} = \frac{r}{\sin\varphi}, \quad \theta\prime = \arcsin\left(\frac{L_2 \sin\varphi}{r}\right)$$

④ Inverse tangent (atan2)

Compute the overall angle α to the point:

目標点への全体角度を計算

$$\alpha = \text{atan2}(y, x)$$

Then the first joint angle is

最終的に第一関節の角度が求まる

$$\theta_1 = \alpha - \theta\prime, \quad \theta_2 = \pi - \varphi$$

What is atan2(y, x)?
A programming function for inverse tangent that returns the correct angle α in the right quadrant so that

$$x = r\cos\alpha, \quad y = r\sin\alpha$$

11

Generalization

I can work out kinematics and inverse kinematics by hand for one-link and two-link robots.

How can we make a general method?

Robot kinematics mostly uses coordinate transformations.
If we describe each transform with a matrix, we get a general theory that applies to various robots.

1リンク・2リンクロボットの順運動学と逆運動学は手計算で求められます。では、これをどうやって一般化できるでしょうか？

ロボットの運動学では座標変換が中心です。これをすべて行列で表すことで、様々なロボットに使える一般的な理論が得られます。

Review of Vectors

A vector is a quantity with both size and direction. In 2D, we write vectors as column vectors to match with matrices later:

ベクトルは、大きさ（長さ）と向きを持つ量

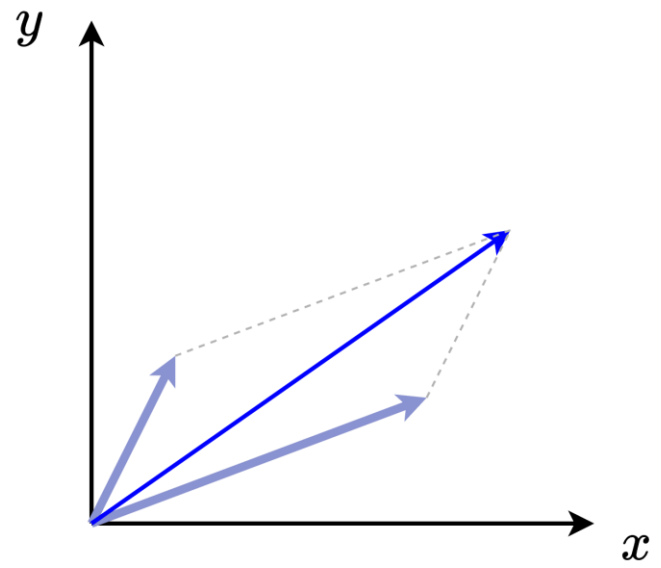
$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

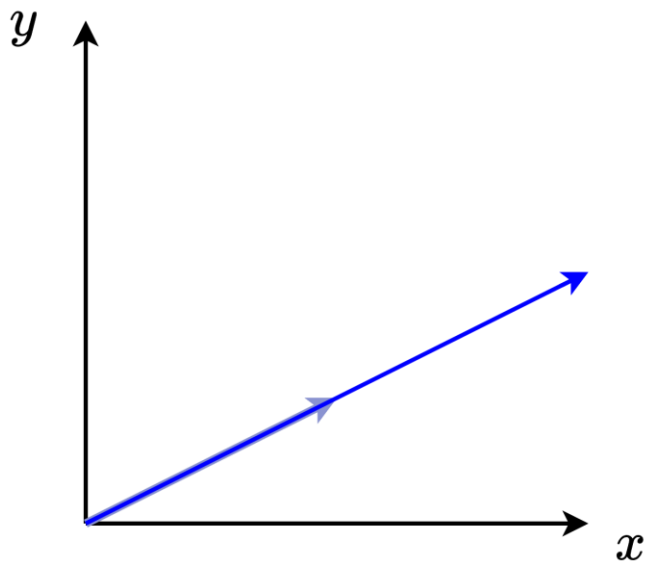
Addition　ベクトルの足し算

$$\mathbf{a} + \mathbf{b} = \begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \end{bmatrix}$$

Scalar Multiplication　スカラー倍

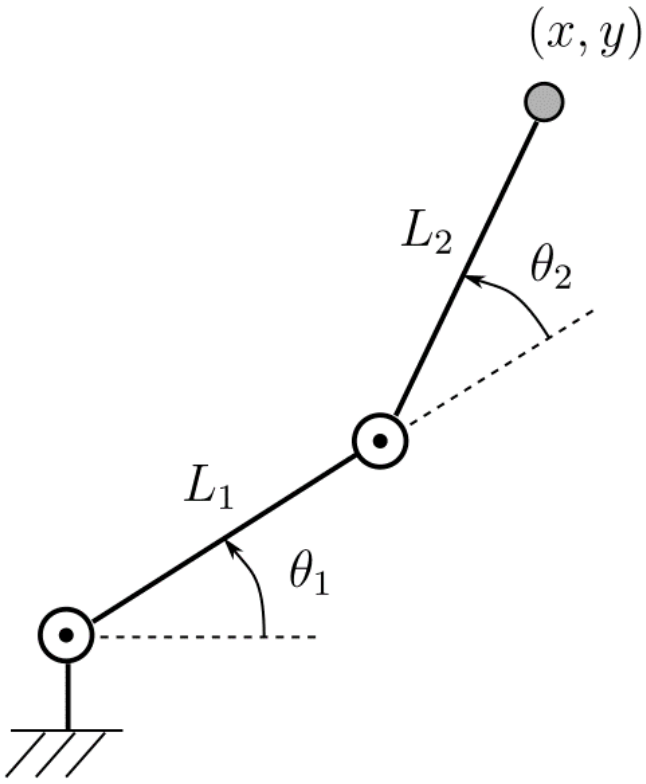
$$k\,\mathbf{a} = \begin{bmatrix} k\,a_1 \\ k\,a_2 \end{bmatrix}$$

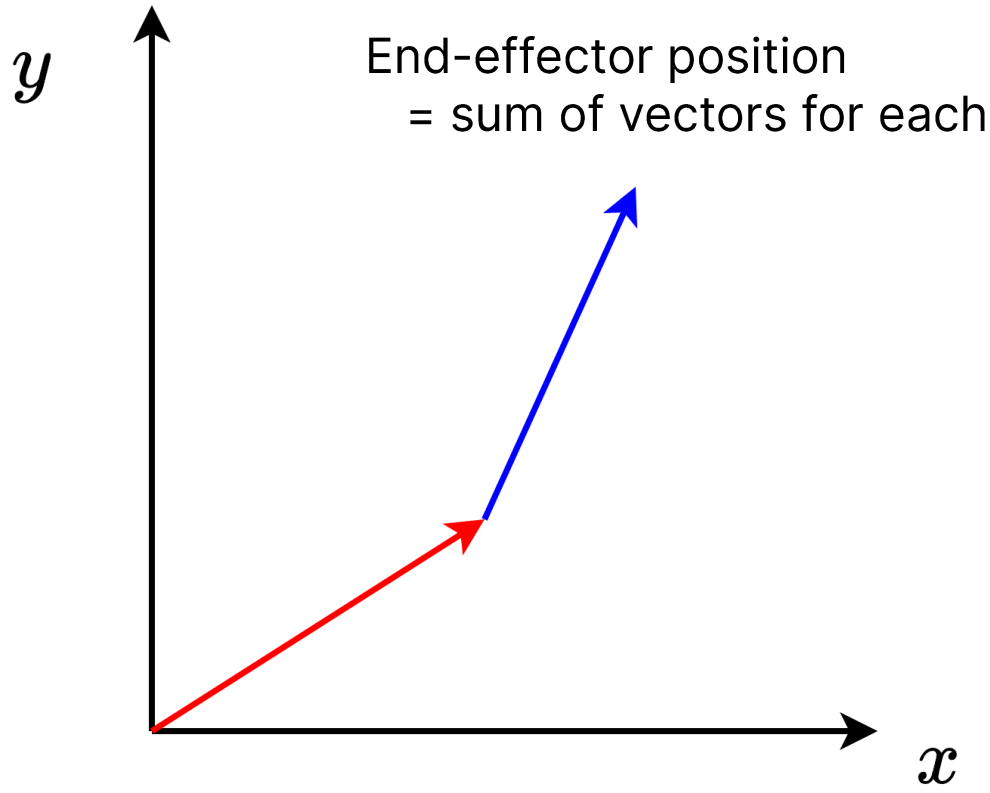




Vectors and Kinematics

If we treat the robot's links as vectors, we can solve kinematics with vector operations.





End-effector position
= sum of vectors for each link.

Matrix 行列

Matrix

- A matrix is an arrangement of numbers in rows and columns.

行列とは、数を行と列に並べたもの

$$\boldsymbol{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

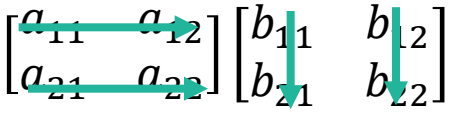
2 x 2

Matrix Addition

- $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$

Matrix Multiplication

- $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$



Matrix

Product of m rows by n columns

- You can multiply two matrices only if the number of columns in the left matrix equals the number of rows in the right matrix.

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} b_{11} & \cdots & b_{1p} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mp} \end{bmatrix}$$

左側の行列の列数と右側の行列の行数が等しい場合にのみ、行列の掛け算ができます

- Then

$$\boldsymbol{C} = \boldsymbol{AB} = \begin{bmatrix} c_{11} & \cdots & c_{1p} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{np} \end{bmatrix}, c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$

- Example:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 1*7+2*9+3*11 & 1*8+2*10+3*12 \\ 4*7+5*9+6*11 & 4*8+5*10+6*12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

Matrix

Characteristics of matrix addition:

- Commutative law $\boldsymbol{A + B = B + A}$ 交換法則
- Associative law $\boldsymbol{A + (B + C) = (A + B) + C}$ 結合法則

Characteristics of matrix multiplication:

- Not commutative $\boldsymbol{AB \neq BA}$ 交換法則は成り立たない
- Associative law $\boldsymbol{A(BC) = (AB)C}$ 結合法則
- Distributive law $\boldsymbol{A(B + C) = AB + AC}$ 分配法則

Matrix and vector multiplication

Matrices can transform vectors using “linear transformations” (also called “mappings”).

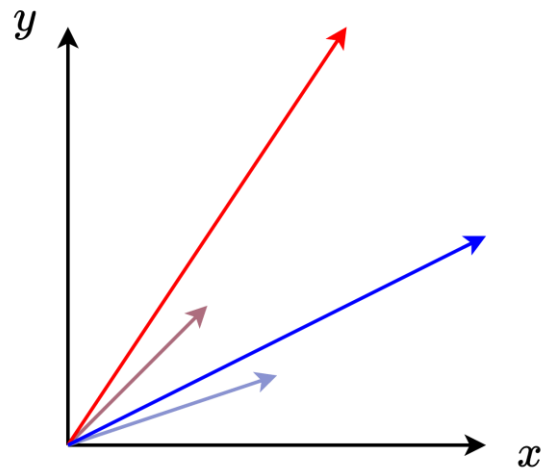
行列は「線形変換」（別名「写像」）を用いてベクトルを変換できます。

- Linear transformation
  - The operation that transforms (x, y) into $(x', y') = (ax + by, cx + dy)$
  - This is represented as $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$, and it is characteristic that it is multiplied from the left side.

after　Mapping matrix　before

Identity matrix (does nothing)　単位行列（何もしない）

- $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$



拡大縮小

Scaling

- $\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2x \\ 3y \end{bmatrix}$
- This example matrix doubles x and triples y.

この例の行列は x を2倍、y を3倍にします。

Rotation Matrix

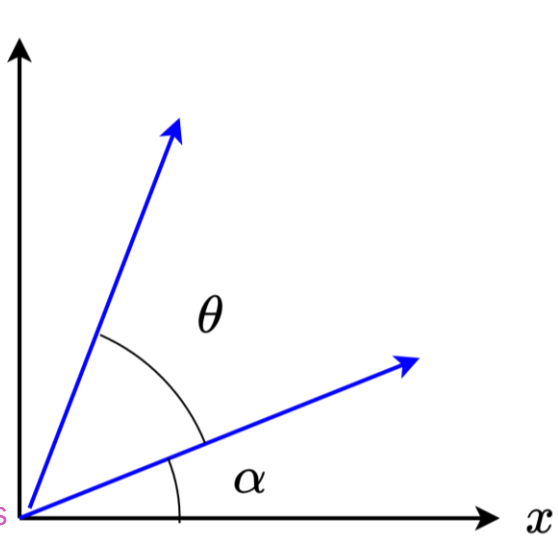
回転行列はロボットの運動学で非常に重要です。
ほとんどのロボット関節は回転関節なので、運動学では回転の計算が多く出てきます。

ベクトル (x, y) を角度 θ だけ回す行列を求める

Rotation matrices are very important in robot kinematics.

- Most robot joints are revolute, so kinematics often involves rotations.

Let's find the matrix that rotates a vector (x, y) by an angle θ



- First, write (x, y) in polar form (using length L and angle α)

$$(x, y) = (L\cos\alpha\,, L\sin\alpha)$$

- Rotating by θ gives

$$\begin{aligned}(x', y') &= (L\cos(\alpha+\theta)\,, L\sin(\alpha+\theta)) \\ &= (L\cos\alpha\cos\theta - L\sin\alpha\sin\theta\,, L\sin\alpha\cos\theta + L\cos\alpha\sin\theta) \\ &= (x\cos\theta - y\sin\theta\,, y\cos\theta + x\sin\theta)\end{aligned}$$

※ using the angle-addition formulas
加法定理を使用

- In matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Example of Combined Transformations

Rotating the vector and then scaling it

- STEP1 Rotation
  - $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$
- STEP2 Scaling
  - $\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$
- Combining them
  - $\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$
    $= \begin{bmatrix} \alpha\cos\theta & -\alpha\sin\theta \\ \beta\sin\theta & \beta\cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

変換の合成例

ベクトルを回転してから拡大縮小する

By using the associative property, we can represent multiple transformations with a single matrix.

結合法則を使うことで、複数の変換を一つの行列で表すことができます。

Rank of matrix

The rank of a matrix is the maximum number of linearly independent column vectors (or row vectors) it has.

- Linearly independent means the vectors point in different directions.

Example

- $\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow \text{rank}(\boldsymbol{A}) = 2$
- $\boldsymbol{B} = \begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix} \rightarrow \text{rank}(\boldsymbol{B}) = 1$

For instance, a transformation with rank = 1 in 2D space makes all transformed vectors lie on a single straight line.

- $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + 2y \\ 3x + 4y \end{bmatrix}$
- $\begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + 2y \\ 3x + 6y \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} (x + 2y)$

行列の階数

行列の階数とは、その行列が持つ線形独立な列ベクトル（または行ベクトル）の最大数です。
• 線形独立とは、ベクトル同士が異なる方向を向いていることを意味します。

たとえば、2次元空間で階数1の変換をすると、変換後のベクトルはすべて一直線上に並びます

Inverse matrix

When a matrix is full rank (its number of rows = its number of columns = its rank), an inverse matrix exists that satisfies:

- $\boldsymbol{AA^{-1} = I}$, $\boldsymbol{I} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$ (Identity matrix)
- The calculation method is omitted here (it's similar to solving a system of equations)

The inverse matrix represents the inverse transformation. For example, for the rotation matrix

- $\boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
- $\boldsymbol{R^{-1}} = \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix}$

行列が正則（行数＝列数＝階数）であるとき、「逆行列」が存在します。
逆行列の求め方は省略します（連立方程式を解く方法に似ています）。
逆行列は、変換の逆を表します。例えば回転行列の逆行列は逆回転を表す

Kinematic Calculation using Matrices and Vectors

Objective

- Find the position and orientation of the robot's end effector relative to the robot's base

Approach

- Use vectors to represent link lengths and linear motions.
- Use rotation matrices to represent joint bends and rotations.
- Define one matrix and one vector for each link.
- Compute the kinematics using matrix and vector operations.

行列とベクトルを用いた運動学計算

目的

- ロボットの基部から見たエンドエフェクタの位置と姿勢を求める。

手法

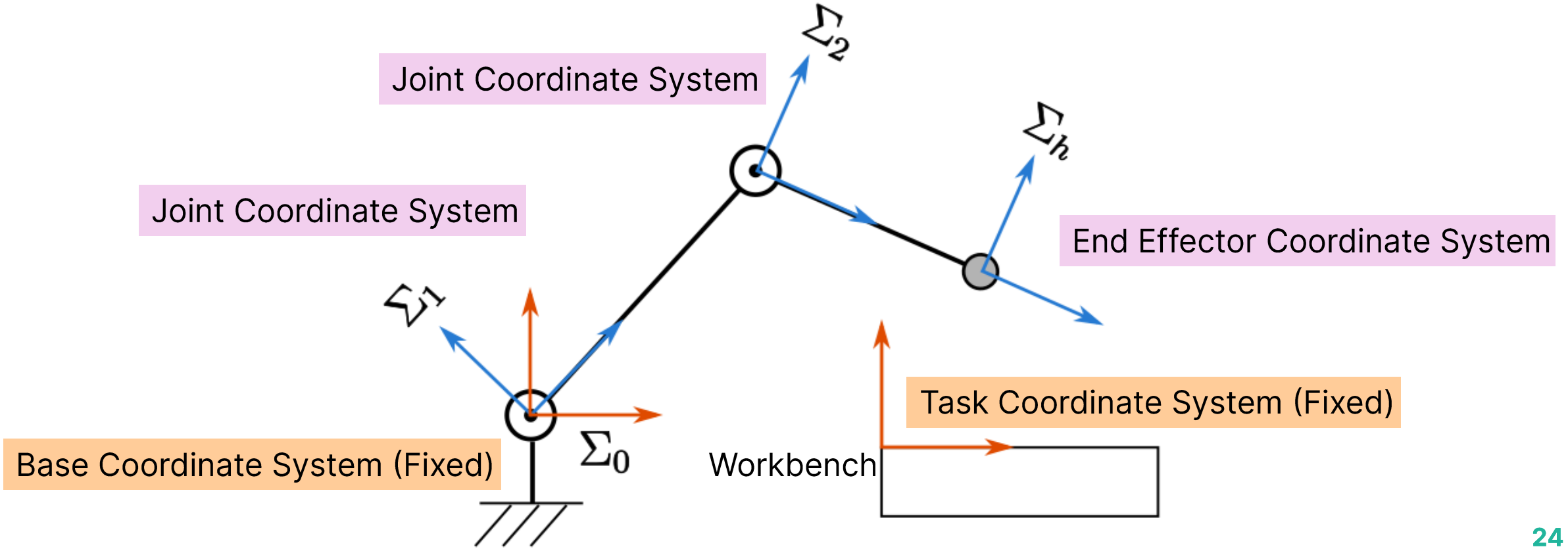
- ベクトルでリンクの長さと直線運動を表現する。
- 回転行列で関節の曲げと回転運動を表現する。
- 各リンクに対して行列１つとベクトル１つを定義する。
- 行列・ベクトル演算で運動学を計算する。

Coordinates

ロボット工学における座標系
- 同じ空間内に複数の座標系が存在します。
- 原点や軸は固定されず、移動・回転することがあります。

Coordinate systems in robotics

- A single workspace can have multiple coordinate systems
- The origins and axes may move or rotate



Vector representation of links

Defining the joint coordinate syste

- The coordinate axes rotate with the joint.
  (See the x- and y-axes on the right.)

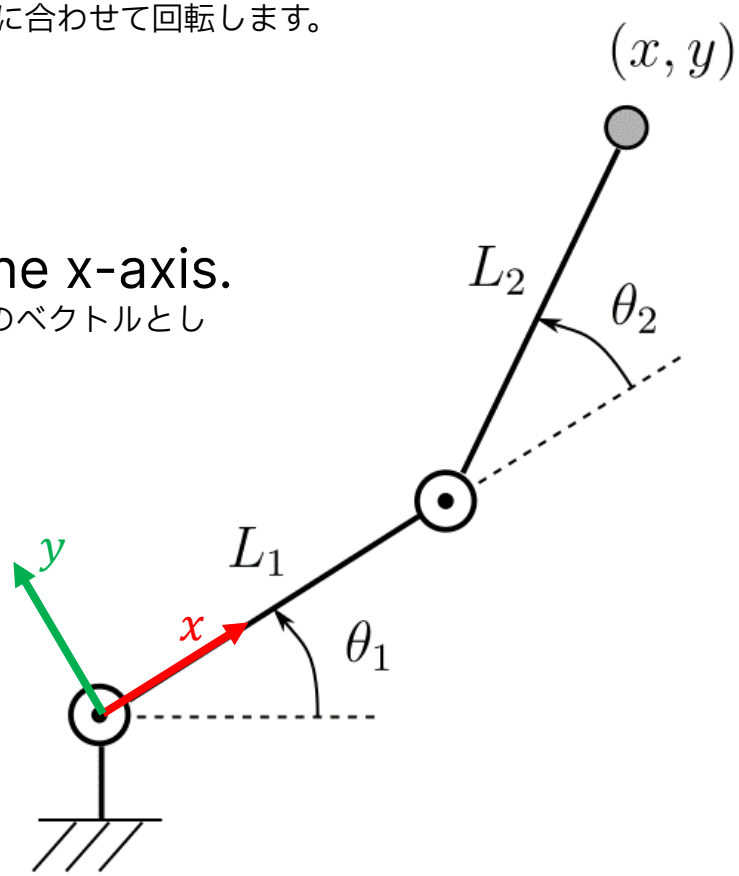
座標軸は関節の回転に合わせて回転します。

Expressing link length as a vector

- We usually represent the link length as a vector along the x-axis.
- In the example on the right,

通常、リンク長は x 軸方向のベクトルとして表現します。

$$\boldsymbol{L}_1 = \begin{bmatrix} L_1 \\ 0 \end{bmatrix}$$



In this case, the forward kinematics result is the sum of:

- $\boldsymbol{L}_1$ viewed from the reference coordinate system
- $\boldsymbol{L}_2$ viewed from the reference coordinate system.

このとき、順運動学の解は以下の合計です。
- 基準座標系で見た L_1
- 基準座標系で見た L_2

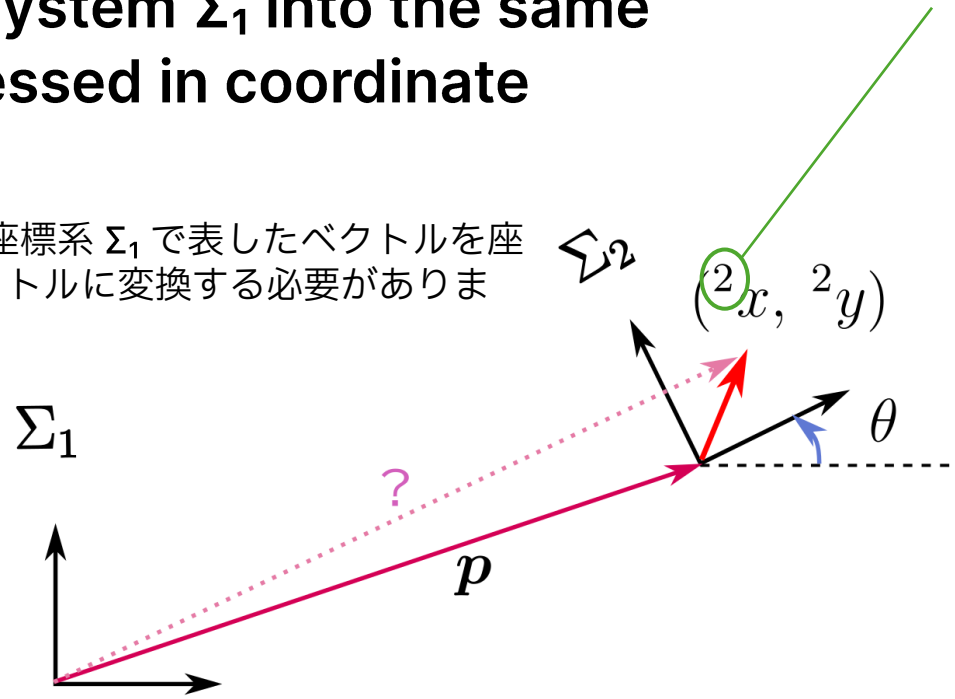
Coordinate Transformation

To solve kinematics, we need to convert a vector expressed in coordinate system Σ_1 into the same vector expressed in coordinate system Σ_2.

運動学を解くには、座標系 Σ_1 で表したベクトルを座標系 Σ_2 で表したベクトルに変換する必要があります。

In robotics, we show which system a vector belongs to by adding a subscript, for example $({}^2x, {}^2y)$ in Σ_2.

ロボット工学では、どの座標系かを示すために添え字を使い、例えば Σ_2 では $({}^2x, {}^2y)$ と書きます。



Given the position relationship (p and θ) between Σ_1 and Σ_2, if p is represented as $({}^2x, {}^2y)$ in Σ_2, how is p written in Σ_1?

Σ_1 と Σ_2 の間で位置関係（p と θ）がわかっているとき、Σ_2 で $({}^2x, {}^2y)$ と表したベクトル p を Σ_1 ではどのように表しますか？

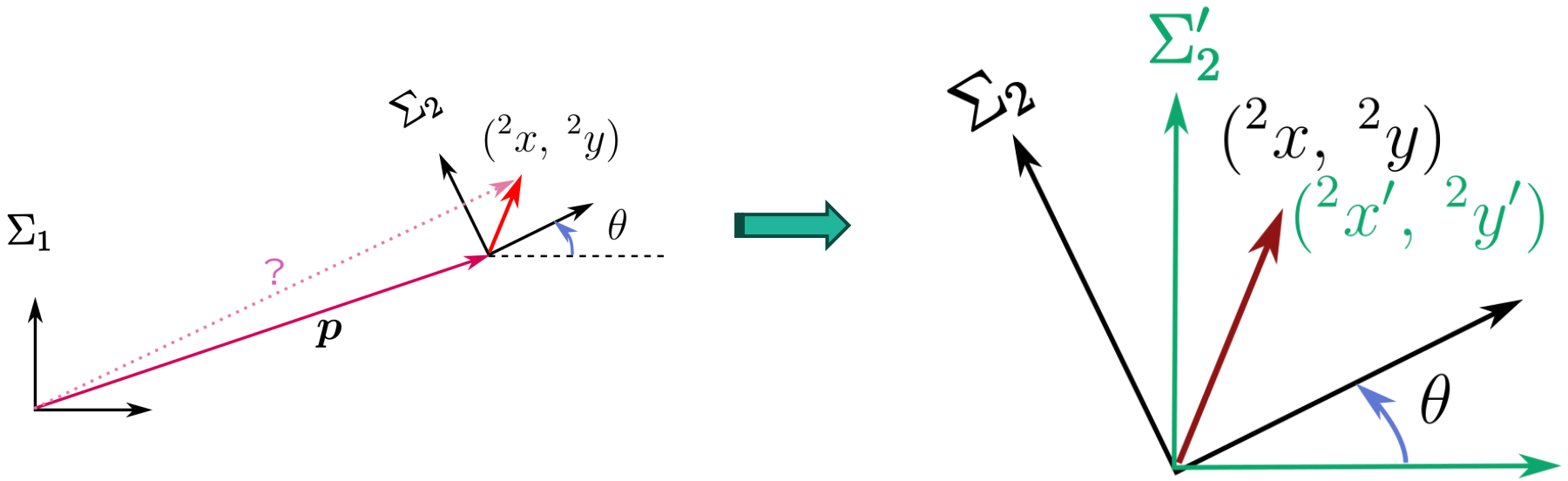
Procedure for Coordinate Transformation 1

First, create a coordinate system Σ_2' that has the same origin as Σ_2 but the same orientation asΣ_1

まず、原点は Σ_2 と同じで、向きは Σ_1 と同じ Σ'_2 座標系を作ります。

Using rotation matrix,

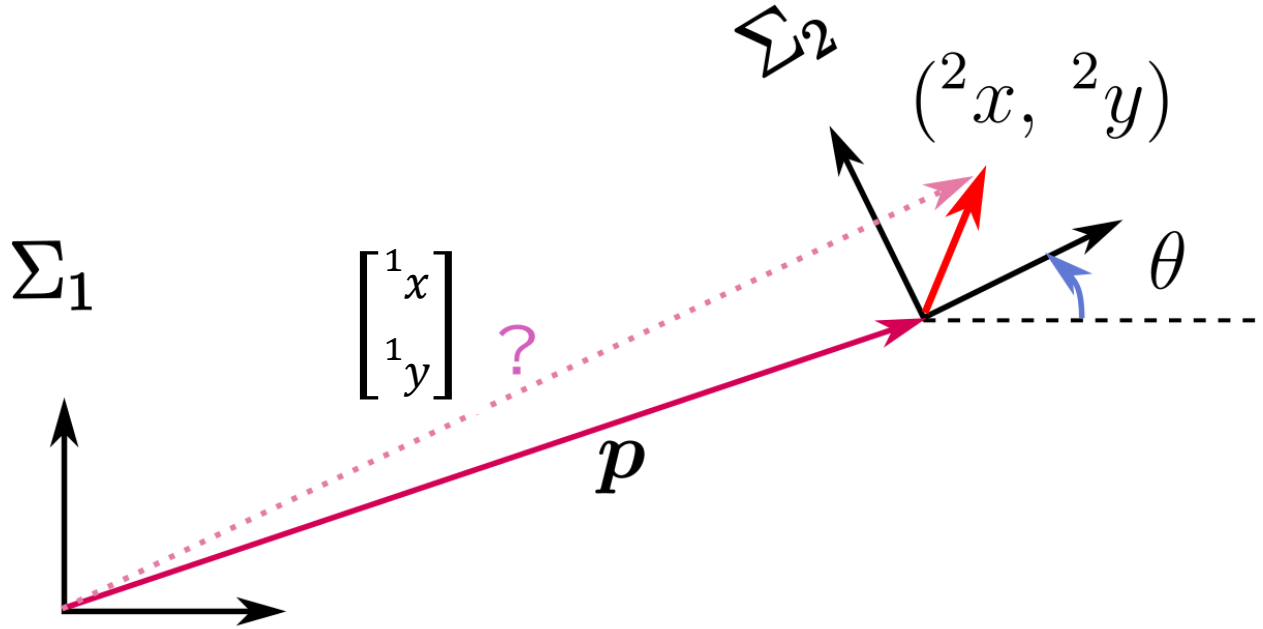
- $\begin{bmatrix} {}^2x' \\ {}^2y' \end{bmatrix} = {}^1\boldsymbol{R}_2 \begin{bmatrix} {}^2x \\ {}^2y \end{bmatrix}, \qquad {}^1\boldsymbol{R}_2 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$



Procedure for Coordinate Transformation 2

Using the formula for the sum of vectors, we can find the final solution $\begin{bmatrix} {}^1x \\ {}^1y \end{bmatrix}$

- $\begin{bmatrix} {}^1x \\ {}^1y \end{bmatrix} = \boldsymbol{p} + {}^1\boldsymbol{R}_2 \begin{bmatrix} {}^2x \\ {}^2y \end{bmatrix}$



Coordinate Conversion Equations

座標変換の式

$$^{1}\boldsymbol{p} = {}^{1}\boldsymbol{p}_{2} + {}^{1}\boldsymbol{R}_{2}\,{}^{2}\boldsymbol{p}$$

- $^{1}\boldsymbol{p}$: Vector p expressed in coordinate system Σ_1
- $^{1}\boldsymbol{p}_{2}$: Position of Σ_2's origin, as a vector in Σ_1
- $^{1}\boldsymbol{R}_{2}$: Rotation matrix from Σ_1 to Σ_2
- $^{2}\boldsymbol{p}$: Vector p expressed in coordinate system Σ_2

Numeric example

Finding ${}^1p$ in the right figure.

- From the figure, you can easily see that ${}^1p = [\,6\,,\ \ 4+\sqrt{2}\,]$, but let's use the coordinate transformation formula to calculate it.

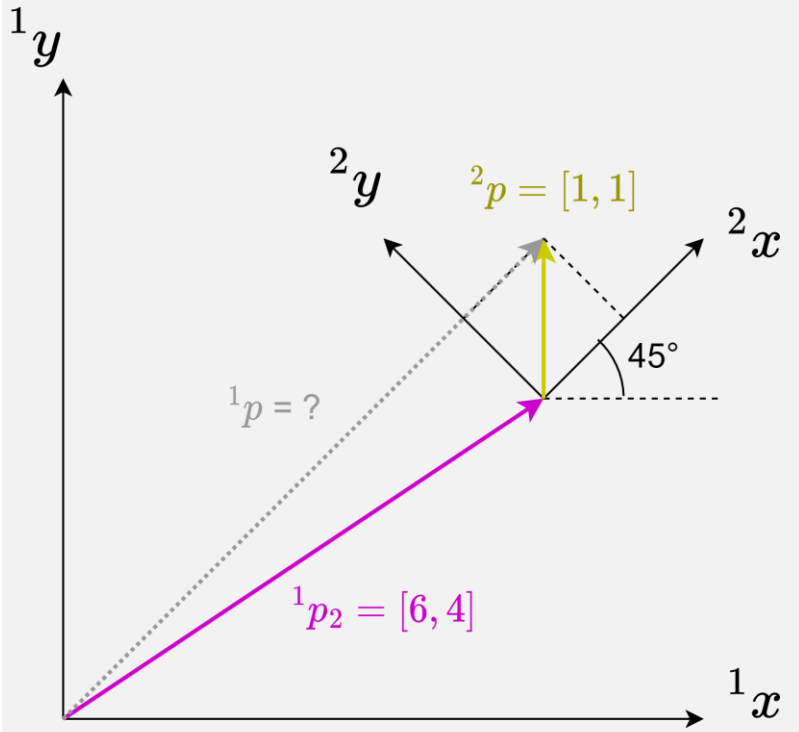
$$ {}^1\boldsymbol{p} = {}^1\boldsymbol{p}_2 + {}^1\boldsymbol{R}_2\,{}^2\boldsymbol{p} $$

$$ {}^1\boldsymbol{R}_2 = \begin{bmatrix} \cos 45° & -\sin 45° \\ \sin 45° & \cos 45° \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} $$

$$ {}^1\boldsymbol{p} = \begin{bmatrix} 6 \\ 4 \end{bmatrix} + \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} $$

$$ = \begin{bmatrix} 6 \\ 4 \end{bmatrix} + \begin{bmatrix} \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} 6 \\ 4+\sqrt{2} \end{bmatrix} $$

図から簡単に ¹p = [6, 4 + √2] とわかりますが、座標変換の式を使って計算してみましょう。

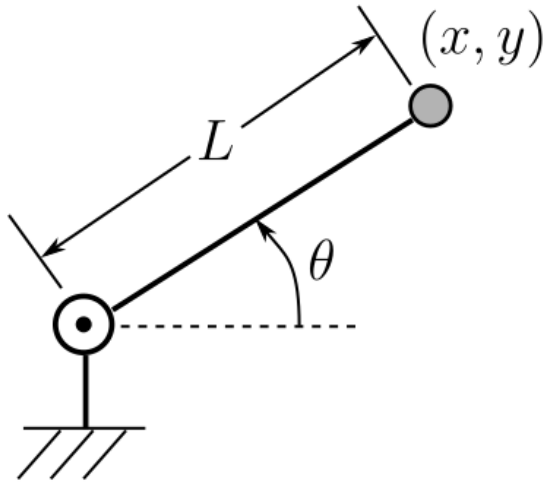


30

Single-link robot example

Kinematics

- $x = L\cos\theta$
- $y = L\sin\theta$



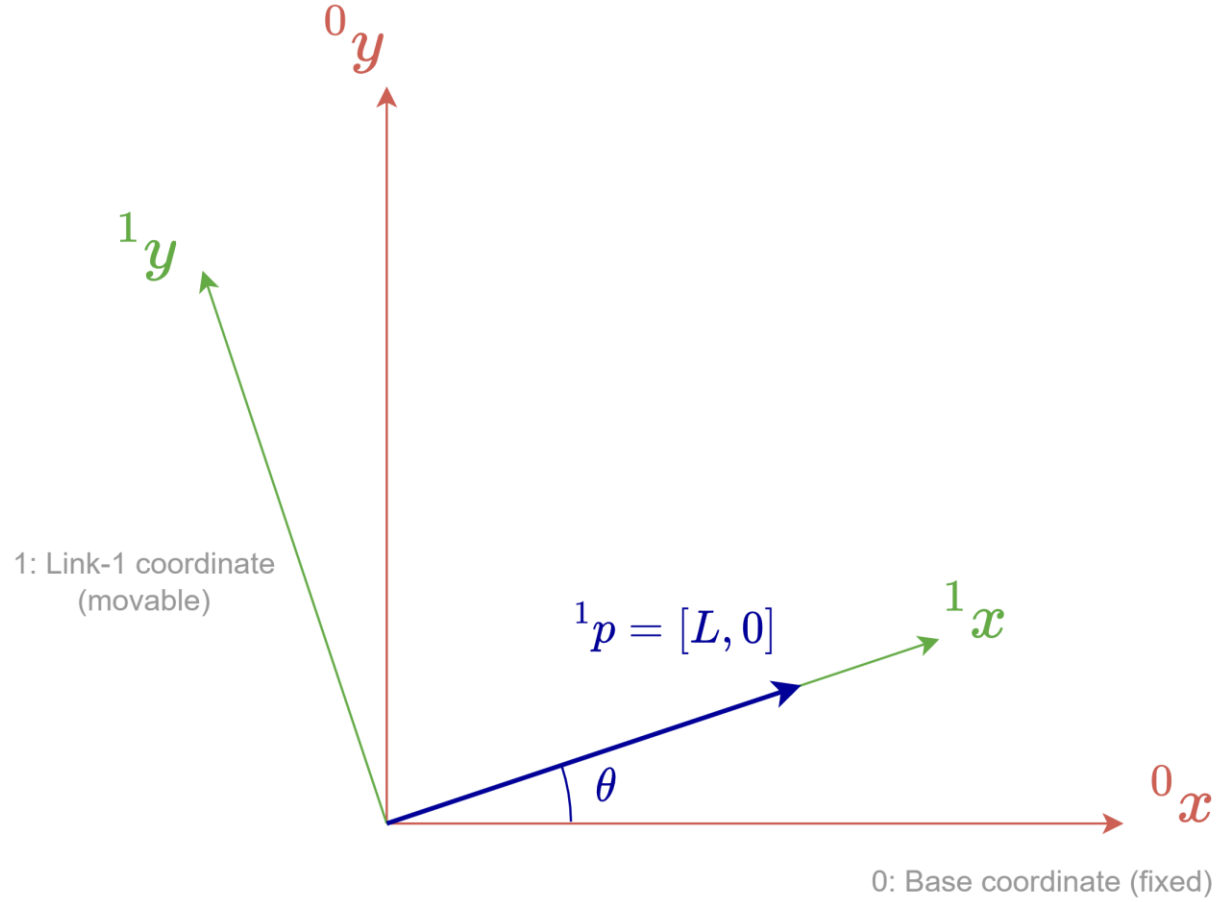
- Let’s derive these kinematic equations using the coordinate transformation formulas.

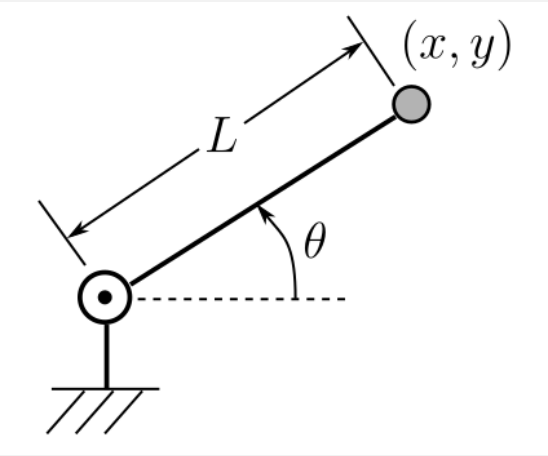
座標変換の式を用いて、上記の運動学を導きましょう。

31

Single-link robot example

Define two coordinate systems as shown in the figure and find ^0p





32

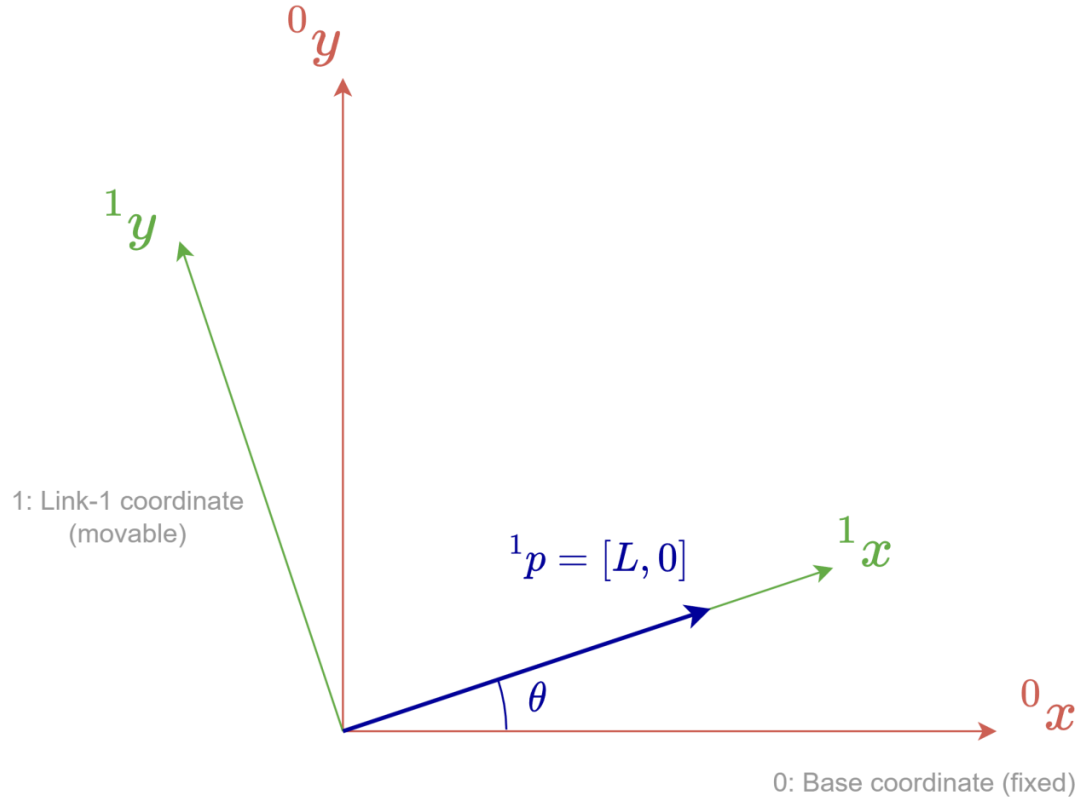
Single-link robot example

$$^0\boldsymbol{p} = {}^0\boldsymbol{p}_1 + {}^0R_1\,{}^1\boldsymbol{p}$$

※

$$= \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}L\\0\end{bmatrix}$$

$$= \begin{bmatrix}L\cos\theta\\ L\sin\theta\end{bmatrix}$$



※ $^0\boldsymbol{p}_1$ is the vector from the origin of frame Σ_0 to the origin of frame Σ_1.
In this example, both frames share the same origin, so $^0p_1 = [0; 0]$.

※ $^0\boldsymbol{p}_1$ は座標系0と座標系1の位置関係を表している。この例では２つの座標系は原点が同じ位置にあるので、$^0\boldsymbol{p}_1$ は零ベクトルになる。

$$^0\boldsymbol{p} = {}^0\boldsymbol{p}_1 + {}^0R_1\,{}^1\boldsymbol{p}$$

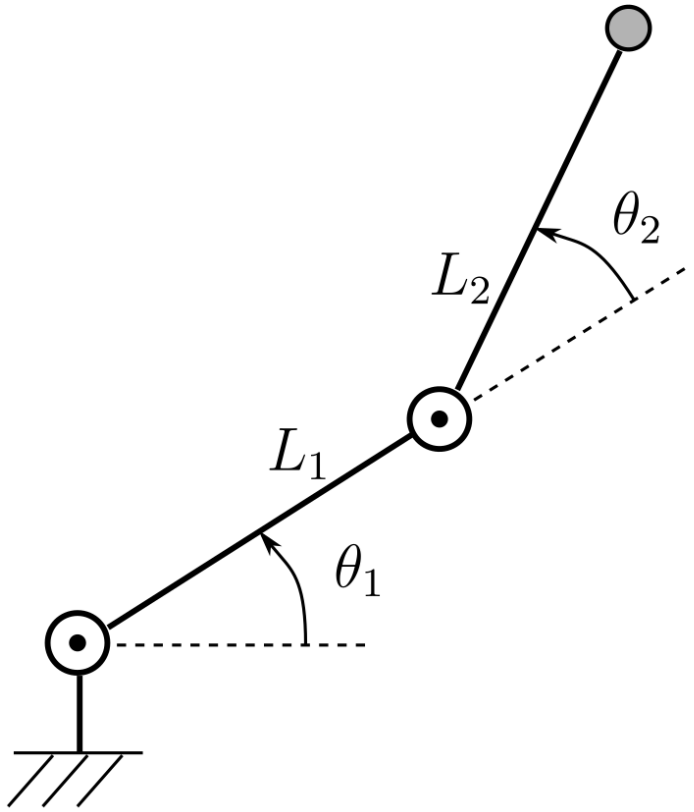
$$= \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}L\\0\end{bmatrix}$$

$$= \begin{bmatrix}L\cos\theta\\ L\sin\theta\end{bmatrix}$$

33

Two-link robot example

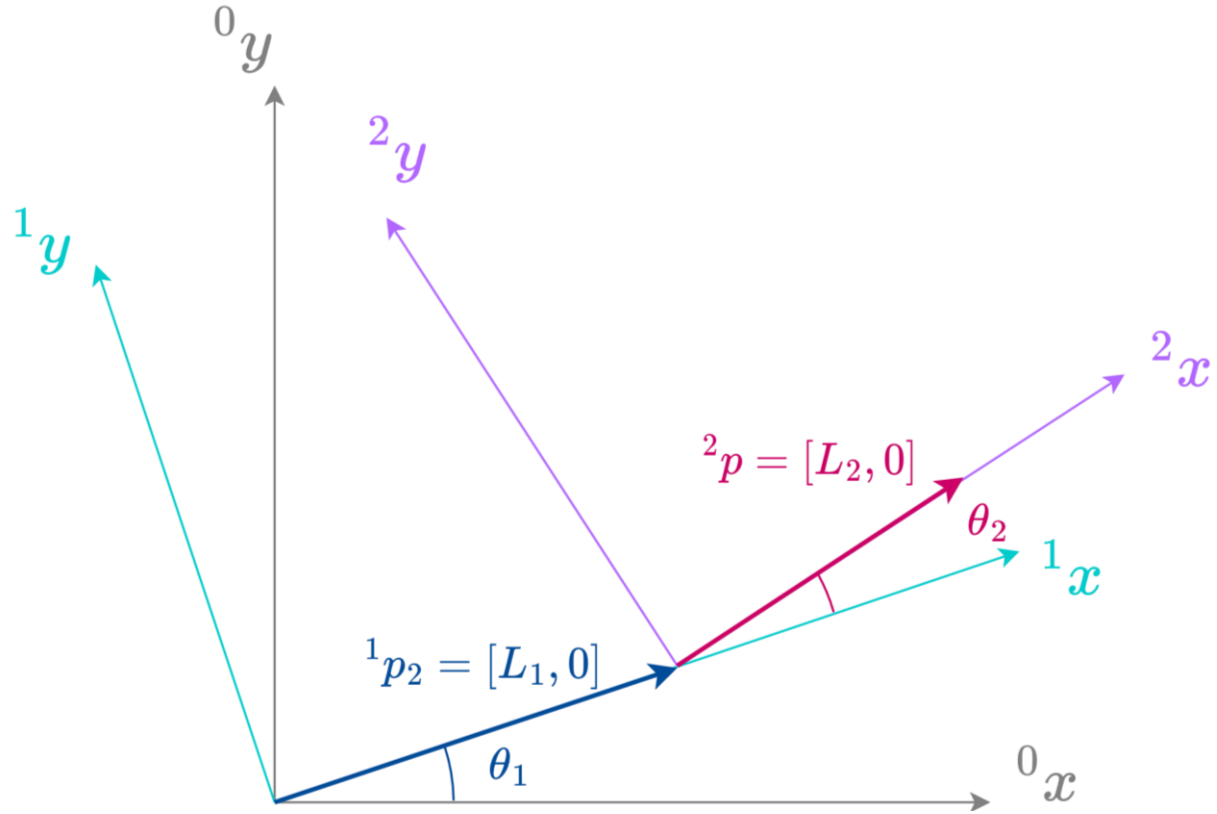
Find the kinematics of the robot arms in the following two links.



34

Two-link robot example

Define the base coordinate system, joint coordinate system 1, and joint coordinate system 2, and find ^0p



基準座標系（Σ_0）、関節座標系1（Σ_1）、関節座標系2（Σ_2）を定義し、^0p を求める。

35

Two-link robot example

$${}^0\boldsymbol{p} = {}^0\boldsymbol{p}_1 + {}^0R_1\,{}^1\boldsymbol{p}$$

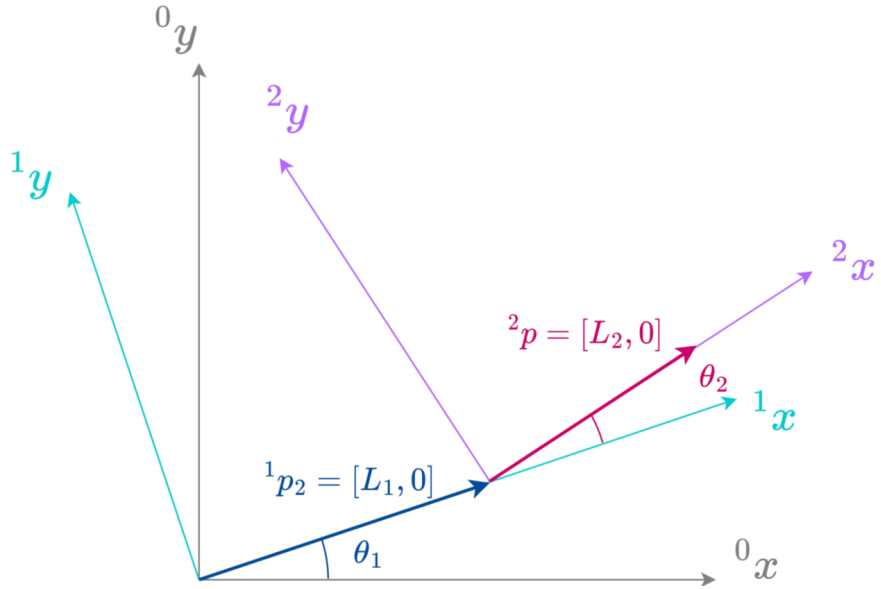
$${}^1\boldsymbol{p} = {}^1\boldsymbol{p}_2 + {}^1R_2\,{}^2\boldsymbol{p}$$

Known variables:

$${}^0\boldsymbol{p}_1 = [0,0],\ {}^0R_1 = R(\theta_1)$$

$${}^1\boldsymbol{p}_2 = [L_1,0],\ {}^1R_2 = R(\theta_2),$$

$${}^2\boldsymbol{p} = [L_2,0]$$



$$\downarrow$$

$${}^1\boldsymbol{p} = \begin{bmatrix} L_1 + L_2\cos\theta_2 \\ L_2\sin\theta_2 \end{bmatrix}$$

$${}^0\boldsymbol{p} = {}^0R_1\,{}^1\boldsymbol{p} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix}\begin{bmatrix} L_1 + L_2\cos\theta_2 \\ L_2\sin\theta_2 \end{bmatrix}$$

$$= \begin{bmatrix} L_1\cos\theta_1 + L_2\cos\theta_1\cos\theta_2 - L_2\sin\theta_1\sin\theta_2 \\ L_1\sin\theta_1 + L_2\sin\theta_1\cos\theta_2 + L_2\cos\theta_1\sin\theta_2 \end{bmatrix} =$$

$$\begin{bmatrix} L_1\cos\theta_1 + L_2\cos(\theta_1+\theta_2) \\ L_1\sin\theta_1 + L_2\sin(\theta_1+\theta_2) \end{bmatrix}$$

36

Homogeneous transformation 同次変換

In a multi-joint robot, we must solve these equations in sequence.

多関節ロボットでは、次の式を順に解く必要があります。

$$
\begin{aligned}
{}^{1}\boldsymbol{p} &= {}^{1}\boldsymbol{p}_{2} + {}^{1}R_{2}\,{}^{2}\boldsymbol{p} \\
{}^{2}\boldsymbol{p} &= {}^{2}\boldsymbol{p}_{3} + {}^{2}R_{3}\,{}^{3}\boldsymbol{p} \\
{}^{3}\boldsymbol{p} &= {}^{3}\boldsymbol{p}_{4} + {}^{3}R_{4}\,{}^{4}\boldsymbol{p} \\
&\vdots
\end{aligned}
$$

Using a homogeneous transformation matrix simplifies this to

これを同次変換行列を使うと、次のように簡潔に表せます

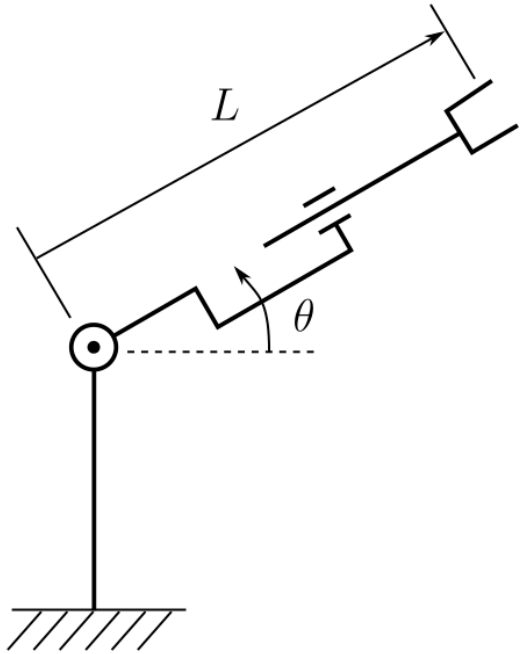
$$
{}^{A}\boldsymbol{T}_{B} = \begin{bmatrix} {}^{A}\boldsymbol{R}_{B} & {}^{A}\boldsymbol{p}_{B} \\ \mathbf{0} & 1 \end{bmatrix}
$$

$$
\begin{bmatrix} {}^{A}\boldsymbol{p} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{A}\boldsymbol{R}_{B} & {}^{A}\boldsymbol{p}_{B} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} {}^{B}\boldsymbol{p} \\ 1 \end{bmatrix}
$$

$$
\Leftrightarrow \quad {}^{A}\boldsymbol{p} = {}^{A}\boldsymbol{p}_{B} + {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{p}
$$

Homogeneous transformation example

Let's find the homogeneous transformation matrix of the following 2-link polar coordinate type robot.



In a prismatic joint, the rotation matrix is the identity matrix.

直動関節では、回転行列は単位行列になります。

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$^{A}T_{B} = \begin{bmatrix} ^{A}R_{B} & ^{A}\boldsymbol{p}_{B} \\ 0 & 1 \end{bmatrix}$$

38

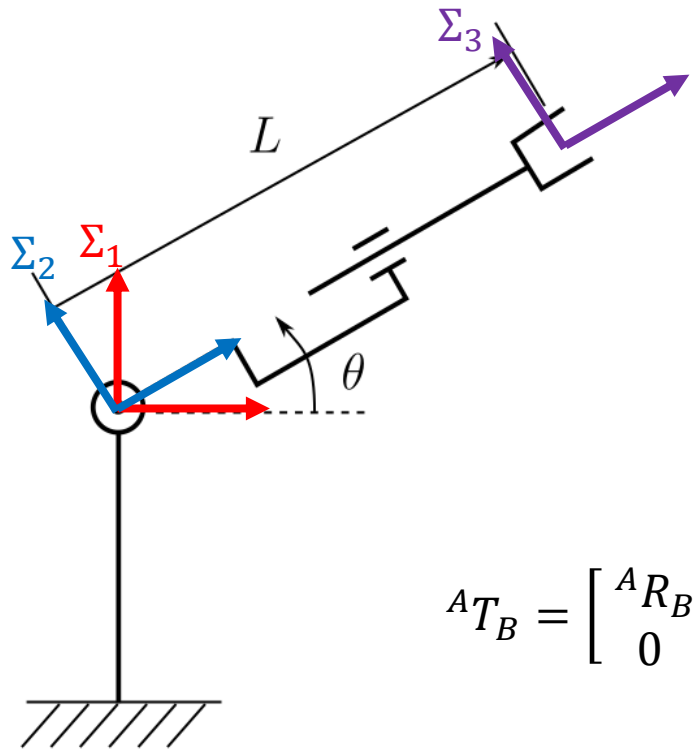
Homogeneous transformation example

${}^0T_1$: **Translation of 0 and rotation by angle** θ

$$\rightarrow {}^0T_1 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

${}^1T_2$: **Translation of** L**, and rotation** 0

$$\rightarrow {}^1T_2 = \begin{bmatrix} 1 & 0 & L \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$${}^A T_B = \begin{bmatrix} {}^A R_B & {}^A \boldsymbol{p}_B \\ 0 & 1 \end{bmatrix}$$

$${}^0T_1\,{}^1T_2 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & L \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & L\cos\theta \\ \sin\theta & \cos\theta & L\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

End effector position

39

Jacobian matrix ヤコビ行列

The Jacobian matrix for a multivariable function $\boldsymbol{y = f(x)}$ is defined as:

多変数関数 y = f(x) のヤコビ行列は次のように定義されます。

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Here ∂ is the partial-derivative operator, which differentiates the function with respect to one variable at a time.

ここで ∂ は偏微分演算子で、関数をある変数に関して微分します。

The Jacobian shows how y changes when each component of x changes slightly.

ヤコビ行列は、x の各成分がわずかに変化したときに y がどのように変化するかを示します。

Example: $z = 2x + 5y$ (equation of a plane)

$$J = \begin{bmatrix} \frac{\partial z}{\partial x} \\ \frac{\partial z}{\partial y} \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

indicating the slope in the x-direction is 2 and in the y-direction is 5.

これは、x 方向の傾きが 2、y 方向の傾きが 5 であることを示しています。

40

Jacobian matrix of a robot

Jacobian Matrix: Describes small changes in output in response to small changes in input.

Robot Jacobian: Relates joint angular velocities to end-effector velocities.

Applications:

- Motion analysis
- Numerical methods for inverse kinematics
- Statics

ヤコビ行列：入力の小さな変化に対する出力の小さな変化を表します。
ロボットのヤコビ行列：関節の角速度とエンドエフェクタの速度の関係を示します。

応用例：
- 運動解析
- 逆運動学などの数値解法
- 静力学

41

Jacobian example

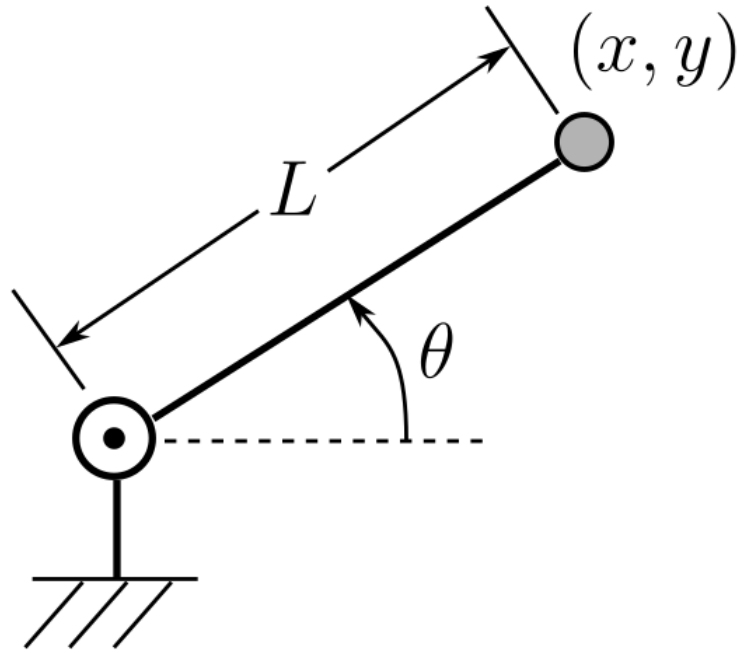
Obtained by differentiating the kinematics.

For a single-link robot:

- $x = L\cos\theta\,,\ y = L\sin\theta$
- $\frac{\partial x}{\partial \theta} = -L\sin\theta\,,\ \frac{\partial y}{\partial \theta} = L\cos\theta$

Thus the Jacobian is

$$J = \begin{bmatrix} -L\sin\theta \\ L\cos\theta \end{bmatrix}$$



- For example, when θ=0, $J = \begin{bmatrix} 0 \\ L \end{bmatrix}$

→ If the joint rotates at 1 rad/s, the end-effector moves along the y-axis at L m/s

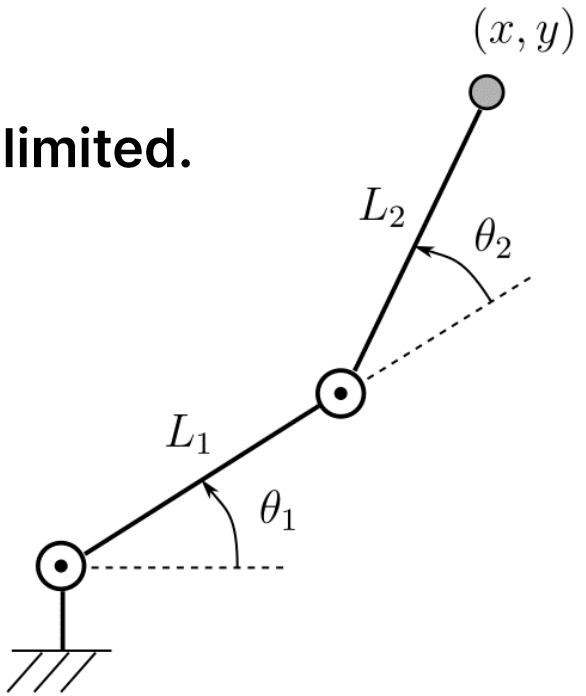
関節が 1 rad/s で回転すると、エンドエフェクタは y 軸方向に L m/s の速度で動きます。

42

Motion Analysis: Rank of Jacobian and Singular Posture

If the rank of the Jacobian matrix is not full, the robot's motion is limited.

ヤコビ行列の階数がフルでないと、ロボットの動きは制限されます。



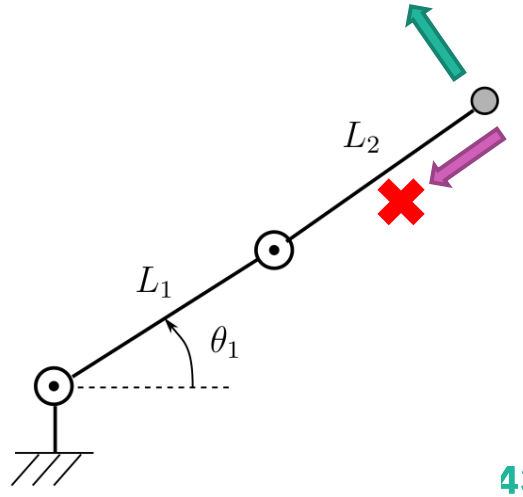
Example (2-link robot)

- $J = \begin{bmatrix} -L_1 \sin\theta_1 - L_2 \sin\theta_{12} & -L_2 \sin\theta_{12} \\ L_1 \cos\theta_1 + L_2 \cos\theta_{12} & L_2 \cos\theta_{12} \end{bmatrix}$

$$(\theta_{12} = \theta_1 + \theta_2)$$

- When $\theta_2 = 0$, which means the arm is fully extended,

$$J = \begin{bmatrix} -L_1 \sin\theta_1 - L_2 \sin\theta_1 & -L_2 \sin\theta_1 \\ L_1 \cos\theta_1 + L_2 \cos\theta_1 & L_2 \cos\theta_1 \end{bmatrix}$$



➔ $\mathrm{rank}(J) = 1$
The end effector of the robot can only move in one direction.

エンドエフェクタは一方向にしか移動できなくなります。

43

Numerical Solution of Inverse Kinematics

Inverse kinematics of a robot often requires the use of numerical methods, as there is no guarantee of an analytical solution.

In this regard, a numerical solution method using the Jacobian matrix as an extension of the Newton-Raphson method will be introduced.

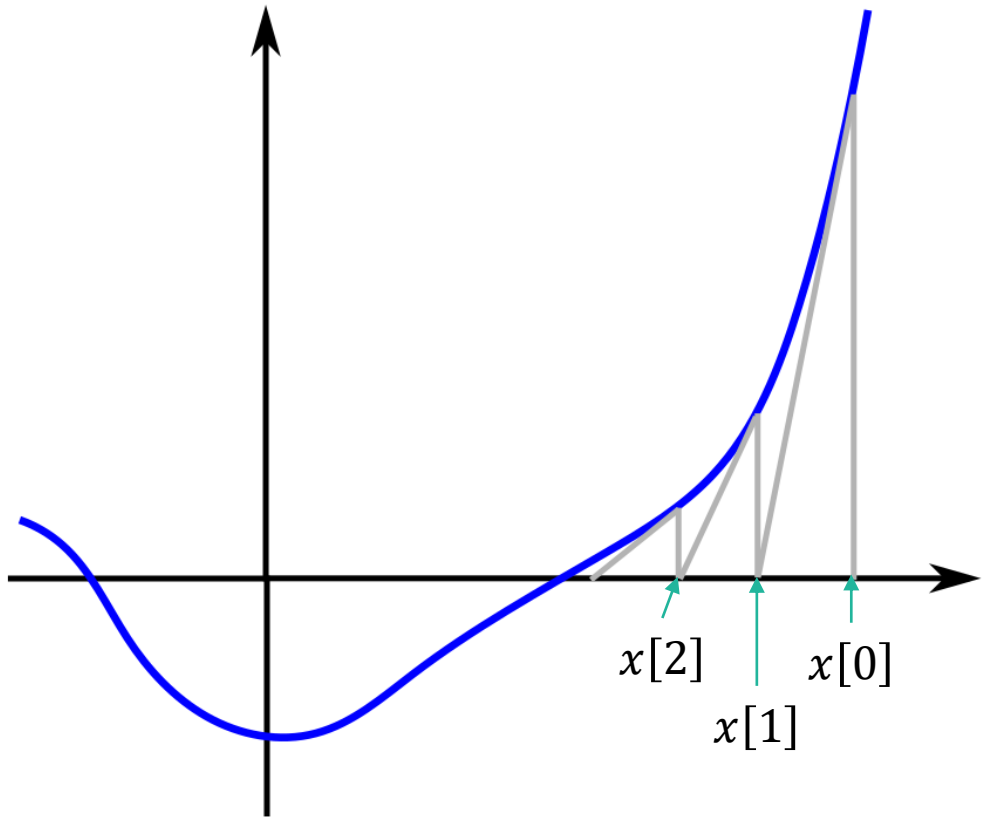
Newton-Raphson method

Problem to solve:

$$f(x) = 0$$

Solution:

$$x[k+1] = x[k] - \beta \frac{f(x)}{f'(x)}$$



44

Newton-Raphson example

$$x[k+1] = x[k] - \beta \frac{f(x)}{f'(x)}$$

Solve $x^2 = 2 \ \ (x > 0)$

$$f(x) = x^2 - 2, \quad f'(x) = 2x$$

$$x[0] = 1, \beta = 1$$

In the original method, there is no parameter β. However, in complex problems such as inverse kinematics, this parameter can be used to tune the performance of numerical solutions.(called damped Newton-Raphson method)

$$x[1] = x[0] - \beta \frac{f(x[0])}{f'(x[0])} = 1 - 1\left(\frac{-1}{2}\right) = 1.5$$

$$x[2] = x[1] - \beta \frac{f(x[1])}{f'(x[1])} = 1.5 - 1\left(\frac{0.25}{3}\right) = 1.4167$$

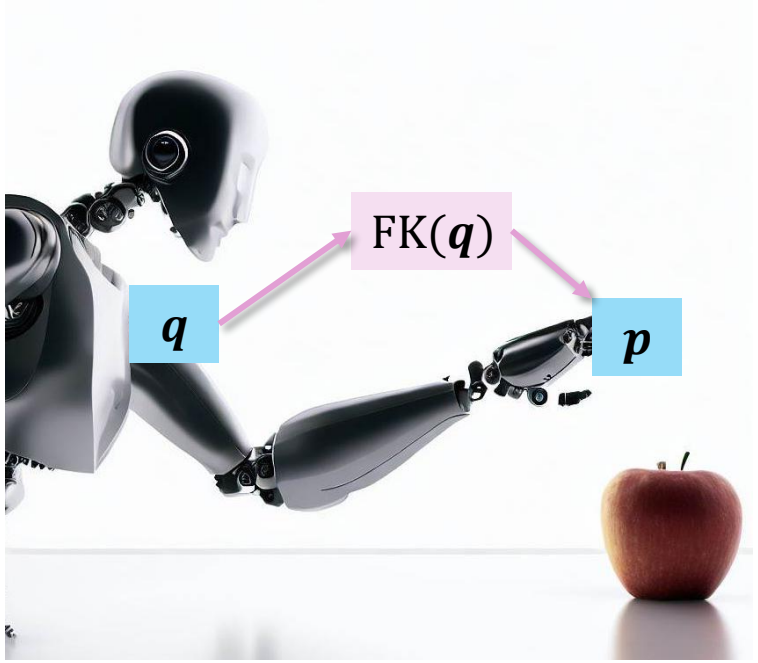
$$x[3] = x[2] - \beta \frac{f(x[2])}{f'(x[2])} = 1.4167 - 1\left(\frac{0.0070}{2.8334}\right) = 1.4142$$

45

Numerical Solution of Inverse Kinematics

Problem setting for inverse kinematics

- Given the end effector position $\boldsymbol{p}$ of a robot, find the joint angles $\boldsymbol{q}$
- $\boldsymbol{p} = \mathrm{FK}(\boldsymbol{q})$ → Solve this equation for $\boldsymbol{q}$
  （FK is forward kinematics）



Method

- Use an iterative update, similar to the Newton–Raphson method.

$$\boldsymbol{q}[k+1] = \boldsymbol{q}[k] + \beta\, J^{-1}\left(\boldsymbol{p} - \mathrm{FK}(\boldsymbol{q}[k])\right)$$

逆運動学の数値解法
【問題設定】
• ロボットのエンドエフェクタ位置 p が与えられたとき、関節角 q を求める。
• p = FK(q) → この式を q について解く（FK は順運動学）。
【手法】
• ヤコビ行列を用いたニュートン–ラフソン法に似た反復計算を実行する：

Newton-Raphson

$$x[k+1] = x[k] - \beta \frac{f(x)}{f'(x)} \quad \text{for} \quad f(x) = 0$$

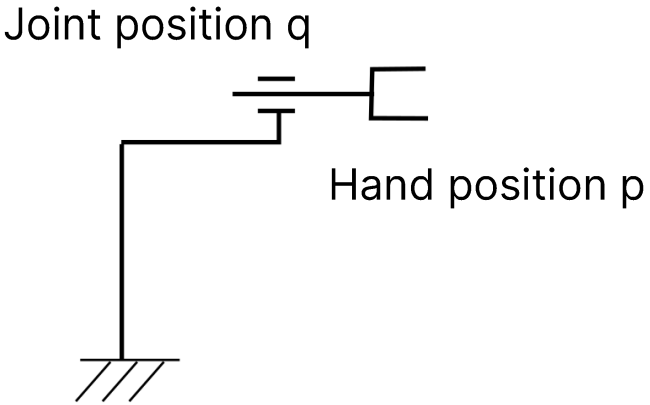
46

Numeric IK example

$$\boldsymbol{q}[k+1] = \boldsymbol{q}[k] + \beta\, J^{-1}\left(\boldsymbol{p} - \mathrm{FK}(\boldsymbol{q}[k])\right)$$

Linear 1-Link robot

- Problem
  - Find q such that $p = 10$
- FK and Jacobian
  - $p = q,\ J = 1$
- Numerical solution parameters
  - $q[0] = 0, \beta = 0.3$



Result (using Excel)

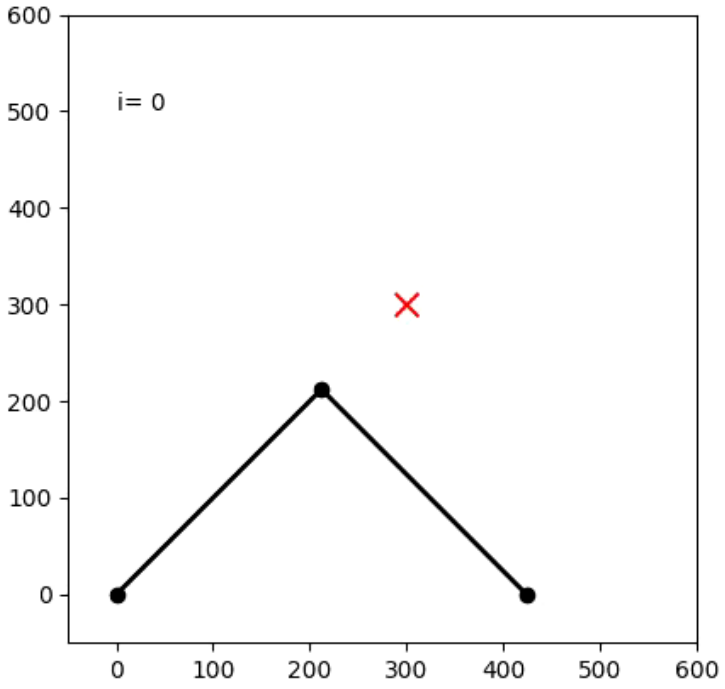
k	q[k]	beta	Jinv	p	FK(q[k])	q[k+1]
0	0	0.3	1	10	0	3
1	3	0.3	1	10	3	5.1
2	5.1	0.3	1	10	5.1	6.57
3	6.57	0.3	1	10	6.57	7.599
4	7.599	0.3	1	10	7.599	8.3193
5	8.3193	0.3	1	10	8.3193	8.82351
6	8.82351	0.3	1	10	8.82351	9.176457
7	9.176457	0.3	1	10	9.176457	9.42352
8	9.42352	0.3	1	10	9.42352	9.596464
9	9.596464	0.3	1	10	9.596464	9.717525
10	9.717525	0.3	1	10	9.717525	9.802267
11	9.802267	0.3	1	10	9.802267	9.861587
12	9.861587	0.3	1	10	9.861587	9.903111
13	9.903111	0.3	1	10	9.903111	9.932178
14	9.932178	0.3	1	10	9.932178	9.952524
15	9.952524	0.3	1	10	9.952524	9.966767
16	9.966767	0.3	1	10	9.966767	9.976737
17	9.976737	0.3	1	10	9.976737	9.983716
18	9.983716	0.3	1	10	9.983716	9.988601

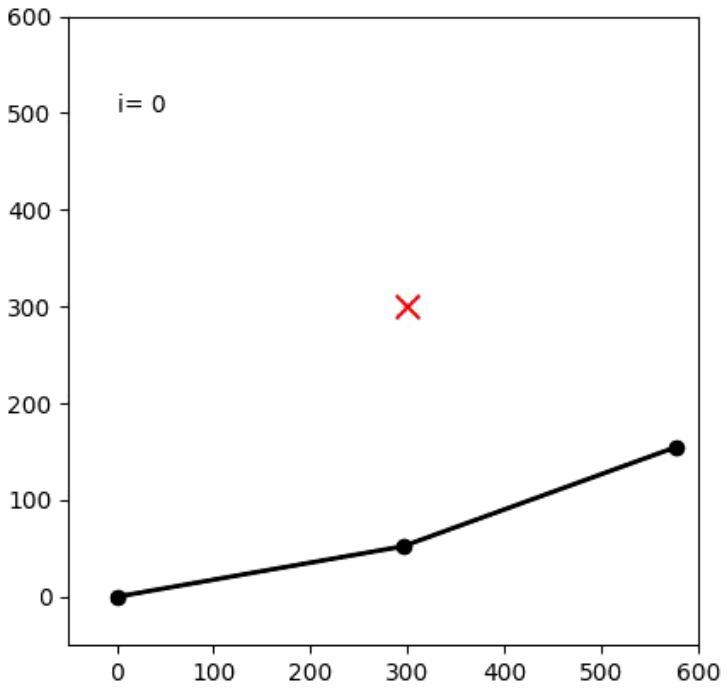
47

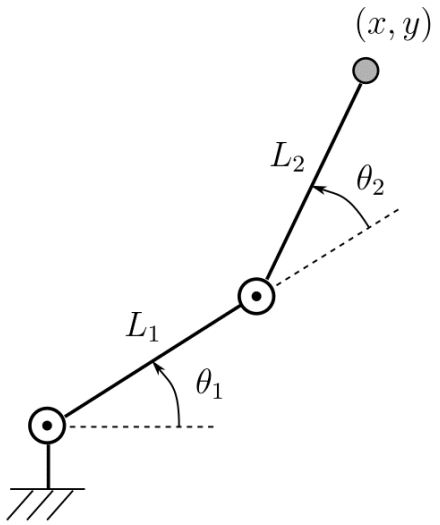
Example 2

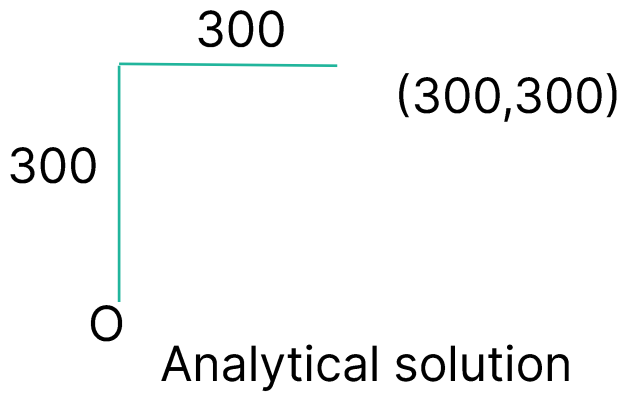
Inverse kinematics of a two-link robot

- L_1 = 300 mm, L_2 = 300 mm









*Choosing an appropriate initial guess is important for obtaining stable solutions.

適切な初期値を選ぶことが、安定した解を得る上で重要です。

48

Python code example

```python
import numpy as np
import matplotlib.pyplot as plt

l_1 = 300
l_2 = 300

def forward_kinematics(q):
    x = l_1 * np.cos(q[0]) + l_2 * np.cos(q[0] + q[1])
    y = l_1 * np.sin(q[0]) + l_2 * np.sin(q[0] + q[1])
    return np.array([x, y])

def jacobian(q):
    J = np.array([[-l_1 * np.sin(q[0]) - l_2 * np.sin(q[0] + q[1]), -l_2 * np.sin(q[0] + q[1])],
                  [l_1 * np.cos(q[0]) + l_2 * np.cos(q[0] + q[1]), l_2 * np.cos(q[0] + q[1])]])
    return J

def draw_links(q, x_d, i):
    x = np.array([0, l_1 * np.cos(q[0]), l_1 * np.cos(q[0]) + l_2 * np.cos(q[0] + q[1])])
    y = np.array([0, l_1 * np.sin(q[0]), l_1 * np.sin(q[0]) + l_2 * np.sin(q[0] + q[1])])

    # clear the plot
    plt.clf()

    # Draw links
    plt.plot(x, y, marker='o', color='k', linestyle='-', linewidth=2)

    # Draw target position
    plt.scatter(x_d[0], x_d[1], s=100, c='red', marker='x')

    # Write the frame number
    plt.text(0, 500, 'i= {}'.format(i), fontsize=10)

    # Set the plot as square
    plt.axis('square')

    # x and y axis range
    plt.xlim(-50, 600)
    plt.ylim(-50, 600)

    # Write animation frames to files
    plt.savefig('frames/{}.png'.format(i))
```

```python
# target position
x_d = np.array([300, 300])

# initial joint angles
q1 = np.deg2rad(45)
q2 = np.deg2rad(-90)

# Create joint angle vector
q = np.array([q1, q2])

# Calculation parameter
beta = 0.3
epsilon = 0.1
i = 0

# Draw initial links
draw_links(q, x_d, i)
i += 1

while True:
    # Calculate forward kinematics
    x = forward_kinematics(q)
    # Calculate Jacobian Inverse
    J_inv = np.linalg.inv(jacobian(q))
    # Use pseudo inverse if inverse is not available
    # J_inv = np.linalg.pinv(jacobian(q))

    # Calculate position error
    e = x_d - x
    # Update joint angle vector
    q = q + beta * np.dot(J_inv, e)

    # Print joint angles
    print('q1: {}, q2: {}'.format(np.rad2deg(q[0]), np.rad2deg(q[1])))

    # Draw links
    draw_links(q, x_d, i)
    i += 1

    # Check if the position error is small enough
    if np.linalg.norm(e) < epsilon:
        break
```

Get source code from:
https://github.com/tak-kanno/robot-lecture

* AI (GitHub Copilot) used in part of the source code.

49

Feedback Control

Using the methods described so far, we can now compute the robot's joint angles for a given desired end-effector position.

How can we drive the motor so that its joint angles match the target values?

Attach an angle sensor to the motor and drive current proportional to the error between the target angle and the measured angle.

→ "feedback control".

フィードバック制御
これまでに説明した手法を用いることで、所望のエンドエフェクタ位置からロボットの関節角度を計算できるようになりました。

では、この目標とする関節角度（ターゲット角度）に合わせてモータをどのように制御すればよいでしょうか？

モータに角度センサを取り付け、目標角度とセンサから得られる実測角度の差（誤差）に比例した電流を流します。
これが「フィードバック制御」です。

P control

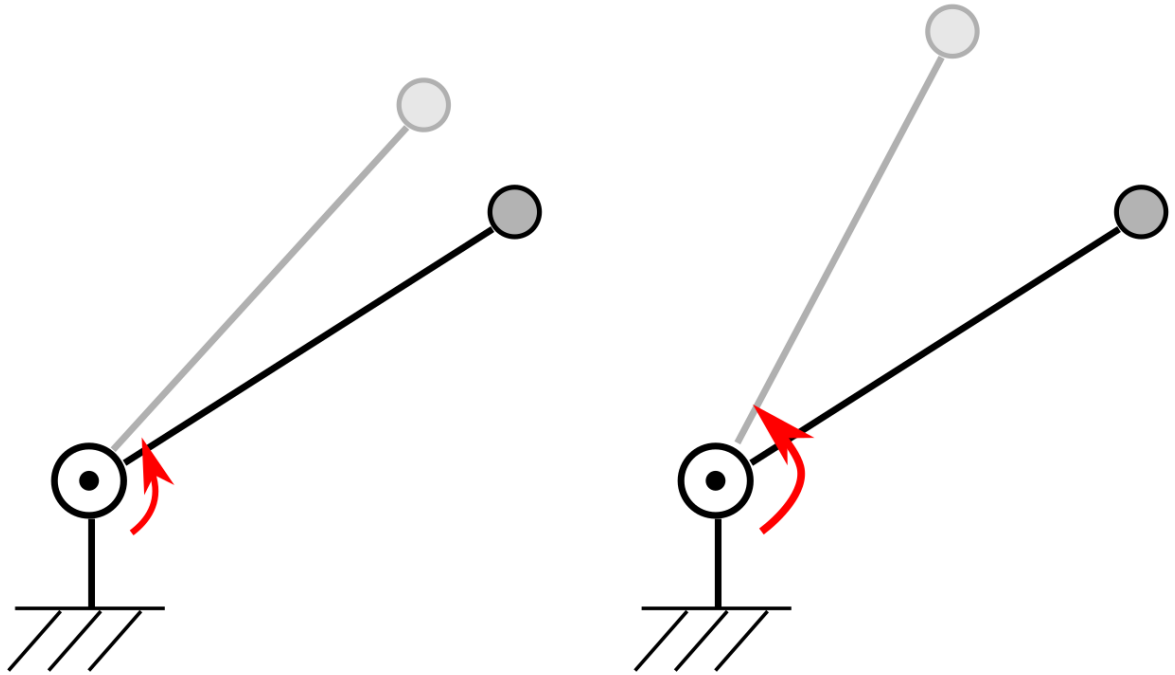
Proportional control

- The motor torque is proportional to the difference between the target angle and the actual angle.

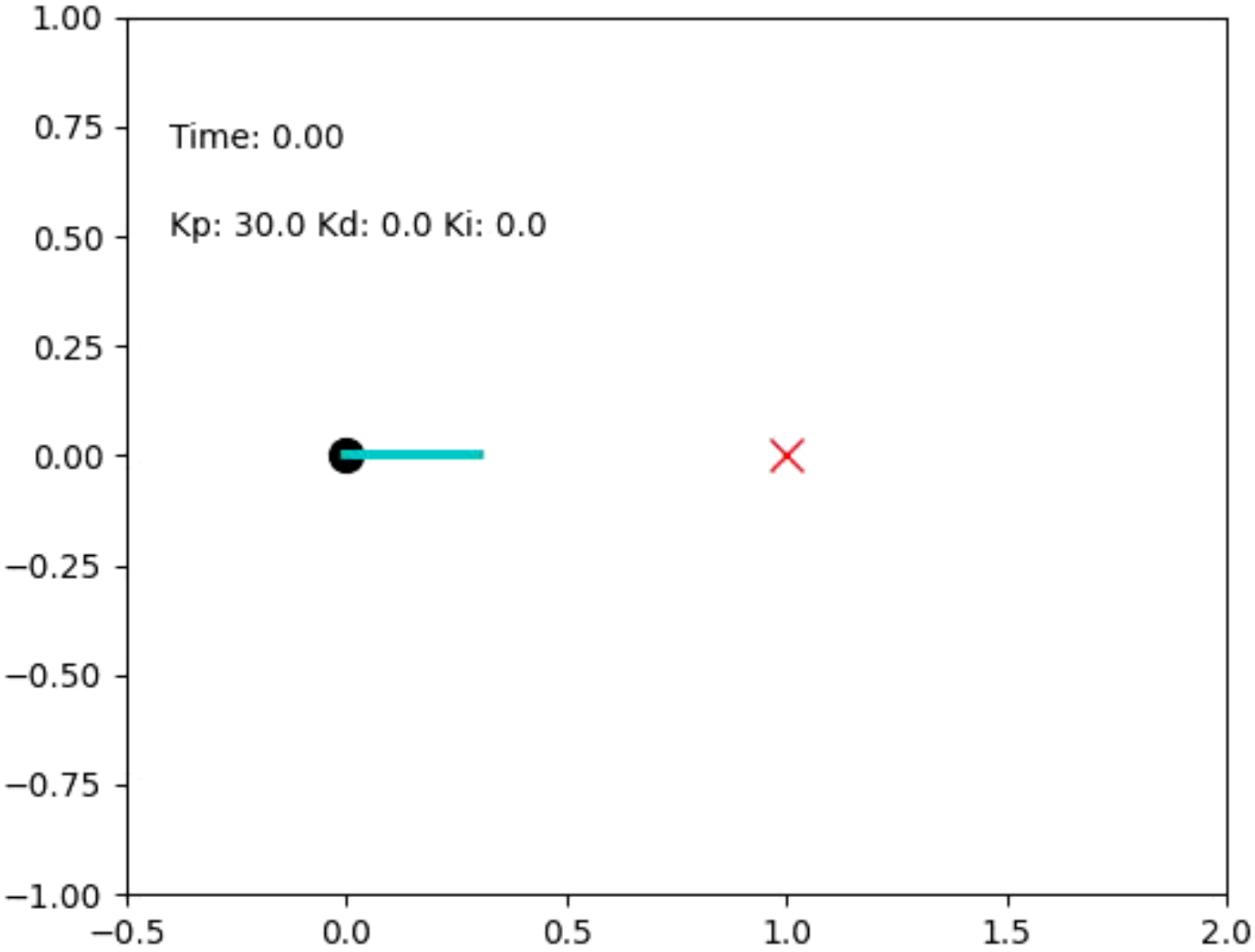
$$\tau = K_{\mathrm{P}}(\theta_{\mathrm{ref}} - \theta)$$

K_{P} ： proportional gain
θ_{ref} ： target angle

- Increasing the gain leads to a fast
likelihood of instability.



P control example



gradually approaching the target value.

D control

Differential control

- The motor torque, which is proportional to the error between the desired angular velocity and the actual angular velocity, is applied.

$$\tau = K_{\mathrm{D}}\left(\dot{\theta}_{\mathrm{ref}} - \dot{\theta}\right)$$

- In D control, when $\dot{\theta}_{\mathrm{ref}} = 0$, $\tau = -K_{\mathrm{D}}\dot{\theta}$, resulting in a resistance force proportional to the velocity. In other words, it functions as a brake and can suppress the vibration caused by P control.

PD control

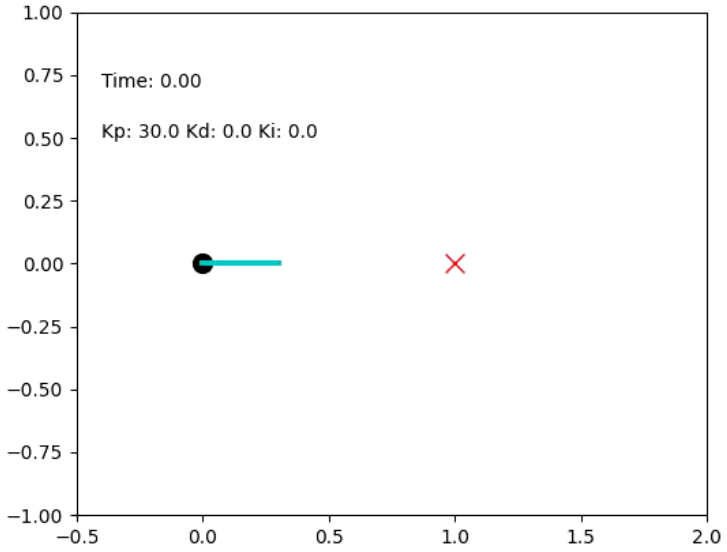
- Combination of P control and D control. Commonly used in feedback control of robots.

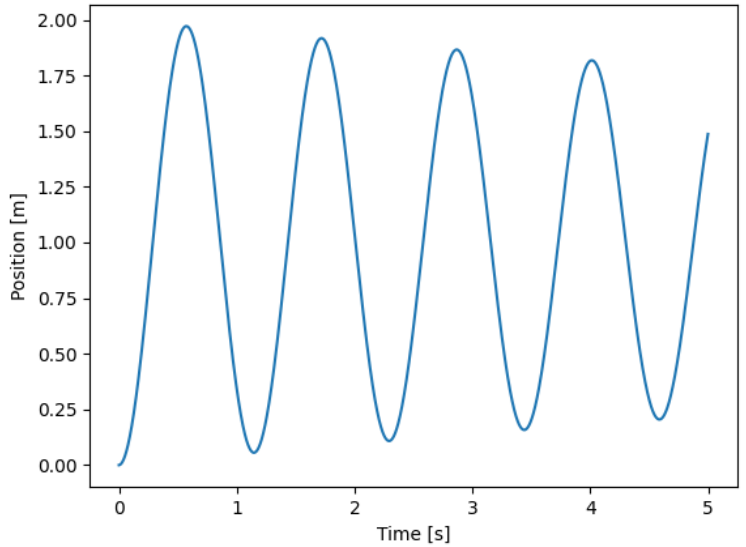
$$\tau = K_{\mathrm{P}}(\theta_{\mathrm{ref}} - \theta) + K_{\mathrm{D}}\left(\dot{\theta}_{\mathrm{ref}} - \dot{\theta}\right)$$

Effect of PD control

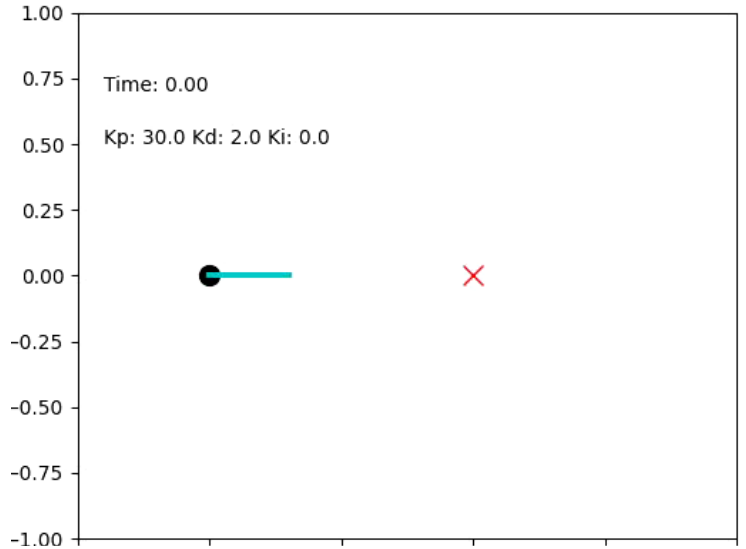
Source code:
https://github.com/tak-kanno/robot-lecture

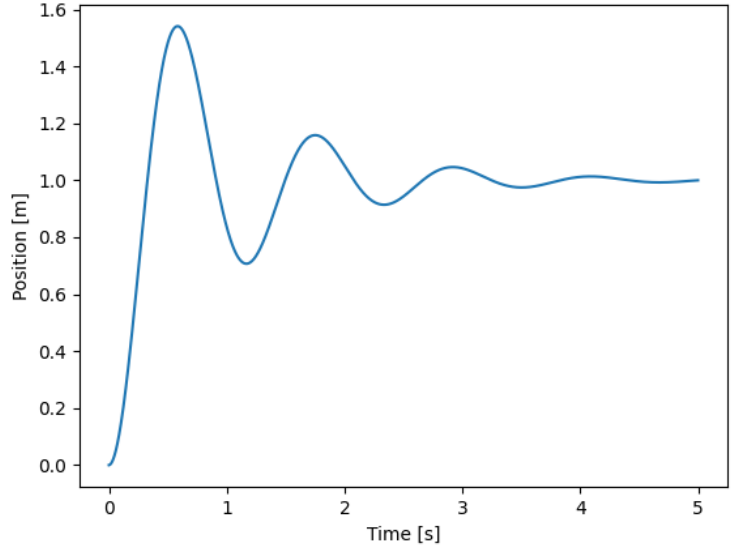
P





PD





Final Assignment (Kanno’s part)

TO BE ANNOUNCED