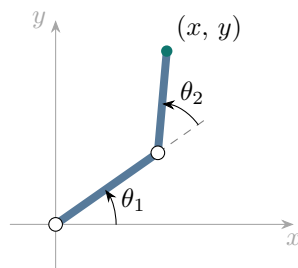


実務につながる

ロボット運動学入門

第I部 位置に関する運動学と軌道生成



初学者のためのテキスト

— 「関節の角度」と「手先の位置・姿勢」を自在に行き来する —

2026 年 8 月 17 日 版

はじめに

ロボットを動かすとき、私たちが最終的に指定したいのは「手先をどこに、どんな向きで持っていくか」です。ところが、実際にモータへ送れる指令は「各関節を何度にするか」だけです。この 2 つの世界——作業座標（手先の位置・姿勢）と関節変位（各関節の角度）——を翻訳する数学が運動学（キネマティクス）です。

- 関節角度 → 手先の位置・姿勢を求める：順運動学（Forward Kinematics; FK）
- 手先の位置・姿勢 → 関節角度を求める：逆運動学（Inverse Kinematics; IK）

本書（第 I 部）は、この翻訳と、その時間的なつながり方である軌道生成までを、初学者向けに一步步説明します。

本書の方針：実務との接続

運動学の教科書は数式が主役になりがちですが、本書では「その式が実機のどこで使われるのか」を常に添えます。各節にある **実務との接続** ボックスでは、産業用ロボットのティーチペンダント、URDF や ROS、ホビー・教育用アーム（シリアルバスサーボで組むアームキットなど）といった現場の道具と、本文の式との対応を紹介します。「この式は、あのボタンの中身だったのか」と思えることを目標にしています。

対象読者と前提知識

- ロボットの運動学に初めて触れる学部生、実機（アームや脚ロボット）をこれから動かしたい人。
- 前提は三角関数、ベクトル、行列の積まで。微分は第 3 章で少しだけ使います。

本書の読み方

各章は次の部品でできています。

- **この章で学ぶこと**：章の入口でゴールを確認します。
- **ポイント**：つまづきやすい所を短くまとめます。
- **例題**：数値を入れて手を動かします。電卓か Python（付録 A）で必ず再現してください。
- **実務との接続**：式と実機の対応です。

- 本章のまとめ：復習用のチェックリストです。

記号の約束

- 角度の単位はラジアンを基本とし、例題では度 ($^{\circ}$) も併記します。
- 略記 $s_1 = \sin \theta_1$, $c_1 = \cos \theta_1$, $s_{12} = \sin(\theta_1 + \theta_2)$, $c_{12} = \cos(\theta_1 + \theta_2)$ を使います。
- ベクトルは太字 $\boldsymbol{p}$ 、行列は大文字 R , T で書きます。
- $\text{atan2}(y, x)$ は第 1 章で説明する「象限を間違えない逆正接」です。

目次

はじめに		iii
第 I 部	位置に関する運動学と軌道生成	1
第 1 章	関節変位と作業座標の関係	3
1.1	平面 2 自由度ロボット	3
1.2	簡単な脚ロボット	6
1.3	垂直型 3 自由度ロボット	7
1.4	本章のまとめ	9
第 2 章	姿勢の記述	11
2.1	姿勢記述のための回転行列	11
2.2	回転行列による座標変換	12
2.3	ロール・ピッチ・ヨー角	13
2.4	ZYZ-オイラー角	14
2.5	単位クォータニオン	14
2.6	本章のまとめ	16
第 3 章	目標軌道の生成	17
3.1	作業座標系での軌道計画（脚ロボット）	17
3.2	関節空間での軌道計画（平面 2 自由度ロボット）	18
3.3	時間の多項式に基づく軌道生成	19
3.4	速度台形則に基づく軌道生成	20
3.5	手先の姿勢に関する軌道計画	21
3.6	単位クォータニオンを使った大円補間	21
3.7	本章のまとめ	22
第 4 章	運動学の一般的表現	25
4.1	リンク座標系と同次変換	25
4.2	リンク座標系の定義とリンクパラメータ	26
4.3	垂直型 3 自由度ロボットの順運動学	27
4.4	垂直型 6 自由度ロボットの順運動学	28
4.5	本章のまとめ	29
第 5 章	実践・位置制御と逆運動学	31
5.1	位置制御されたモータによって駆動されるロボット	31
5.2	垂直型 6 自由度ロボット：手先一軸の制御	32
5.3	大まかな場所を実現するアーム部	33

5.4	本章のまとめ	35
付録 A	数値計算で確かめる (Python)	37
参考文献・さらに学ぶために		39

第Ⅰ部 位置に関する運動学と軌道 生成

第 1 章 関節変位と作業座標の関係

この章で学ぶこと

- もっとも簡単なロボット（平面 2 自由度）で、順運動学と逆運動学を自分の手で導けるようになる。
- 同じ考え方が脚ロボットにもそのまま使えることを知る。
- 3 次元の垂直型ロボットが「回る平面の中の 2 リンク」に分解できることを理解する。

ロボットの構造はさまざまですが、運動学の考え方の核はごく単純なモデルで全部学べます。この章では、平面の 2 自由度アーム、脚ロボットの 1 本の脚、垂直型の 3 自由度アームという 3 つのモデルを通して、「関節変位 → 作業座標」（順運動学）と「作業座標 → 関節変位」（逆運動学）の往復を練習します。

1.1 平面 2 自由度ロボット

モデルの設定

図 1.1 のように、長さ l_1 , l_2 の 2 本のリンクが回転関節でつながったアームを考えます。根元の関節角を θ_1 (x 軸から測る)、肘の関節角を θ_2 (リンク 1 の延長線から測る) とし、手先の位置を (x, y) とします。

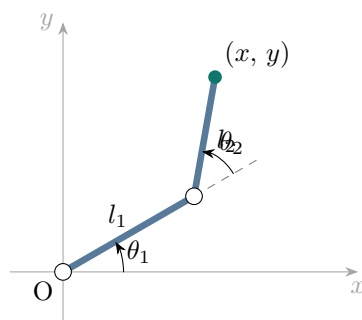


図 1.1 平面 2 自由度ロボット。関節が 2 つ（白丸）、手先が 1 点（緑丸）。

順運動学：角度から位置へ

リンク 1 の先端は $(l_1 c_1, l_1 s_1)$ にあります。リンク 2 は x 軸から測って角度 $\theta_1 + \theta_2$ を向いているので、その先端 (= 手先) は

$$\boxed{x = l_1 c_1 + l_2 c_{12}, \quad y = l_1 s_1 + l_2 s_{12}} \quad (1.1)$$

です。これが平面 2 自由度ロボットの**順運動学**です。角度を決めれば位置は一意に、しかも四則演算と三角関数だけで求まります。

例題 1.1 順運動学の計算

$l_1 = 0.30 \text{ m}$, $l_2 = 0.25 \text{ m}$, $\theta_1 = 30^\circ$, $\theta_2 = 60^\circ$ のときの手先位置を求めよ。

解答 $\theta_1 + \theta_2 = 90^\circ$ なので

$$x = 0.30 \cos 30^\circ + 0.25 \cos 90^\circ = 0.260 + 0 = 0.260 \text{ m},$$

$$y = 0.30 \sin 30^\circ + 0.25 \sin 90^\circ = 0.150 + 0.250 = 0.400 \text{ m}.$$

手先は $(0.260, 0.400) \text{ m}$ にあります。

ポイント：atan2 とは

逆運動学では「sin と cos の両方から角度を復元する」操作が繰り返し登場します。関数 $\text{atan2}(y, x)$ は、点 (x, y) の偏角を $(-\pi, \pi]$ の範囲で返す関数で、 $\tan^{-1}(y/x)$ と違って**象限を取り間違えません** ($x = 0$ でも破綻しません)。たとえば $\text{atan2}(1, -1) = 135^\circ$ ですが、 $\tan^{-1}(1/(-1)) = -45^\circ$ になってしまいます。プログラムでは必ず **atan2** を使ってください。ほぼすべての言語に標準で入っています。

逆運動学：位置から角度へ

今度は手先の目標位置 (x, y) が与えられたとき、 θ_1, θ_2 を求めます。原点から手先までの距離を $r = \sqrt{x^2 + y^2}$ とおきます。図 1.1 の三角形 O-A-手先 に余弦定理を使うと、リンク間の内角は $\pi - \theta_2$ なので

$$r^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos(\pi - \theta_2) = l_1^2 + l_2^2 + 2l_1 l_2 \cos \theta_2.$$

これを $\cos \theta_2$ について解くと

$$\cos \theta_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}, \quad \sin \theta_2 = \pm \sqrt{1 - \cos^2 \theta_2}, \quad \theta_2 = \text{atan2}(\sin \theta_2, \cos \theta_2). \quad (1.2)$$

$\sin \theta_2$ の符号 $\pm$ が**肘の曲げ方向** (肘上げ／肘下げ) に対応します。 θ_2 が決まれば、 θ_1 は「目標方向の角度」から「リンク 1 と手先方向のずれ」を引いて

$$\theta_1 = \text{atan2}(y, x) - \text{atan2}(l_2 \sin \theta_2, l_1 + l_2 \cos \theta_2). \quad (1.3)$$

例題 1.2 逆運動学の計算

$l_1 = 0.30 \text{ m}$, $l_2 = 0.25 \text{ m}$ のロボットで、手先を $(x, y) = (0.40, 0.20) \text{ m}$ に置きたい。関節角を求めよ。

解答 式 (1.2) より

$$\cos \theta_2 = \frac{0.40^2 + 0.20^2 - 0.30^2 - 0.25^2}{2 \cdot 0.30 \cdot 0.25} = \frac{0.0475}{0.15} = 0.3167.$$

$\sin \theta_2 = +\sqrt{1 - 0.3167^2} = 0.9485$ (肘下げ側) を選ぶと $\theta_2 = \text{atan2}(0.9485, 0.3167) = 71.5^\circ$ 。式 (1.3) より

$$\theta_1 = \text{atan2}(0.20, 0.40) - \text{atan2}(0.25 \times 0.9485, 0.30 + 0.25 \times 0.3167) = 26.6^\circ - 32.0^\circ = -5.5^\circ.$$

検算：式 (1.1) に $(\theta_1, \theta_2) = (-5.5^\circ, 71.5^\circ)$ を代入すると $(x, y) = (0.400, 0.200)$ に戻ります。なお $\sin \theta_2$ を負に選べば、鏡写しのもう 1 つの解 $(\theta_1, \theta_2) = (58.6^\circ, -71.5^\circ)$ が得られます。

作業領域と解の個数

式 (1.2) が解を持つのは $|\cos \theta_2| \leq 1$ のとき、すなわち

$$|l_1 - l_2| \leq r \leq l_1 + l_2 \quad (1.4)$$

のときだけです。手先が届く範囲（作業領域）は図 1.2 のようなドーナツ状の領域になります。

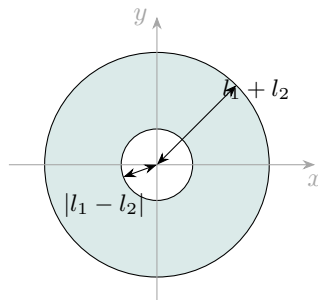


図 1.2 平面 2 自由度ロボットの作業領域（薄い緑の部分）。

ポイント：逆運動学の 3 つの顔

順運動学の答えは常に 1 つですが、逆運動学は (i) 解が 2 つある（肘上げ／肘下げ）、(ii) 解が 1 つ（作業領域の境界上）、(iii) 解がない（届かない）、の 3 通りが起きます。「複数解からどれを選ぶか」「届かないときどうするか」は、実装で必ず向き合う問題です（第 5 章）。

実務との接続

平面 2 自由度の式 (1.1), (1.2), (1.3) は、そのまま実機で使われています。

- **スカラ (SCARA) ロボット**：電子部品の実装や搬送で定番の水平多関節ロボットは、水平面内がまさにこの 2 リンクです。
- **ホビー・教育用アーム**：サーボモータ 2 個と工作材料で作る「お絵かきアーム」も同じ式です。本章の 2 つの式を実装するだけで、狙った座標にペン先を置けます。
- 垂直多関節ロボットの肩・肘（第 1.3 節）も、平面を切り出せばこの 2 リンクに帰着します。まずこの節を完全に理解するのが最短ルートです。

1.2 簡単な脚ロボット

アームと脚は一見別物ですが、運動学としては同じです。図 1.3 のように、胴体に付いた股関節（角度 θ_1 ）と膝関節（角度 θ_2 ）を持つ 2 自由度の脚を考えます。脚では「胴体から見て足先がどこにあるか」が知りたいので、股関節を原点に、前方を x 、鉛直下向きを z とする座標系を使い、 θ_1 は鉛直下向きから前方への傾きで測ります。

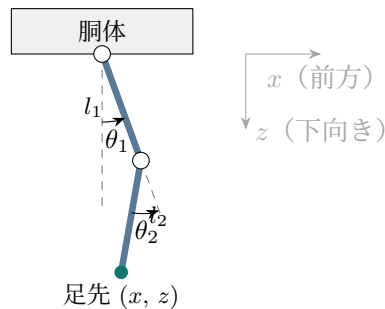


図 1.3 2 自由度の脚。股関節 H と膝関節 K。 θ_1 は鉛直下向きから、 θ_2 は太もも（リンク 1）の延長線から測る。

順運動学

「 x 軸から測る」代わりに「鉛直下向き（ z 軸）から測る」だけなので、式 (1.1) の $\cos$ と $\sin$ が入れ替わった形になります。

$$x = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2), \quad z = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2). \quad (1.5)$$

$\theta_1 = \theta_2 = 0$ のとき足先は真下 $(0, l_1 + l_2)$ に来ることを確かめてください。

逆運動学

距離の式は前節と同一なので、 θ_2 は式 (1.2) の y を z に置き換えるだけです。

$$\cos \theta_2 = \frac{x^2 + z^2 - l_1^2 - l_2^2}{2l_1 l_2}. \quad (1.6)$$

θ_1 は「鉛直下向きから足先方向への角度」から補正角を引いて

$$\theta_1 = \text{atan2}(x, z) - \text{atan2}(l_2 \sin \theta_2, l_1 + l_2 \cos \theta_2) \quad (1.7)$$

となります (atan2 の引数の順番が前節と入れ替わっていることに注意。下向き基準だからです)。

脚でも肘 (膝) の曲げ方向で 2 つの解が出ますが、実機の脚は**膝の曲がる向きが機構的に決まっている**のが普通です。「膝は前に曲がる ($\theta_2 > 0$)」のように符号を規約で固定すれば、解は 1 つに定まります。

例題 1.3 足先の配置

$l_1 = l_2 = 0.15 \text{ m}$ の脚で、足先を股関節の 5 cm 前方、25 cm 下方、すなわち $(x, z) = (0.05, 0.25) \text{ m}$ に置きたい。膝は $\theta_2 > 0$ 側に曲がるとして関節角を求めよ。

解答

$$\cos \theta_2 = \frac{0.05^2 + 0.25^2 - 0.15^2 - 0.15^2}{2 \cdot 0.15 \cdot 0.15} = \frac{0.020}{0.045} = 0.444, \quad \theta_2 = 63.6^\circ.$$

$\sin \theta_2 = 0.896$ を使い、式 (1.7) より

$$\theta_1 = \text{atan2}(0.05, 0.25) - \text{atan2}(0.15 \times 0.896, 0.15 + 0.15 \times 0.444) = 11.3^\circ - 31.8^\circ = -20.5^\circ.$$

つまり太ももを後ろへ 20.5° 傾け、膝を 63.6° 曲げれば、足先は指定位置に来ます。式 (1.5) での検算も一致します。

実務との接続

- **歩行ロボットの足配置**：4 足歩行ロボットの制御では、「次の一步で足先を胴体基準のどこへ着くか」をまず作業座標で決め、それを本節の逆運動学で関節角に翻訳します。この計算を 4 本の脚それぞれについて毎制御周期 (数ミリ秒ごと) 行うのが歩行制御の土台です。
- 実際の脚は股関節に横方向の自由度 (ロール) も持ちますが、その場合も「まず横の傾きを決め、残りを本節の 2 自由度で解く」という分解がよく使われます。

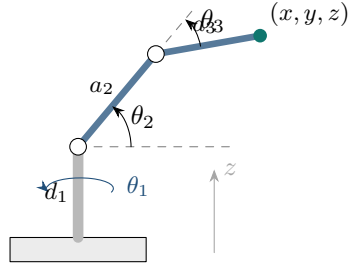
1.3 垂直型 3 自由度ロボット

いよいよ 3 次元です。図 1.4 のような、工場でおなじみの垂直多関節型の最初の 3 軸を考えます。

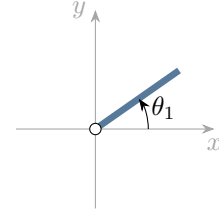
- 第 1 関節 (腰)：鉛直軸まわりの回転 θ_1 。アーム全体の向きを変える。

- 第2関節（肩）：水平軸まわりの回転 θ_2 （水平から上向きに測る）。
- 第3関節（肘）：水平軸まわりの回転 θ_3 （リンク2の延長から測る）。

土台から肩までの高さを d_1 、上腕・前腕の長さを a_2, a_3 とします。



(a) 側面から見た図



(b) 真上から見た図

図 1.4 垂直型 3 自由度ロボット。 θ_1 でアームの乗る鉛直平面ごと回り、その平面内は 2 リンクである。

順運動学：「回る平面の中の 2 リンク」

このロボットのうまい見方は、「 θ_2, θ_3 の 2 リンクが乗った鉛直平面が、 θ_1 で丸ごと回る」というものです。平面内での「肩からの水平距離 ρ 」と「肩からの高さ ζ 」は、前節までとまったく同じ 2 リンクの式で

$$\rho = a_2 c_2 + a_3 c_{23}, \quad \zeta = a_2 s_2 + a_3 s_{23}$$

です。この平面が z 軸まわりに θ_1 回っているので、

$$x = c_1 (a_2 c_2 + a_3 c_{23}), \quad y = s_1 (a_2 c_2 + a_3 c_{23}), \quad z = d_1 + a_2 s_2 + a_3 s_{23} \quad (1.8)$$

が順運動学になります。

例題 1.4 垂直型 3 自由度の順運動学

$d_1 = 0.10, a_2 = 0.30, a_3 = 0.25 \text{ m}, (\theta_1, \theta_2, \theta_3) = (30^\circ, 45^\circ, -30^\circ)$ のときの手先位置を求めよ。

解答 $\theta_2 + \theta_3 = 15^\circ$ 。まず平面内で

$$\rho = 0.30 \cos 45^\circ + 0.25 \cos 15^\circ = 0.212 + 0.241 = 0.454, \quad \zeta = 0.30 \sin 45^\circ + 0.25 \sin 15^\circ = 0.212 + 0.065 = 0.277.$$

よって

$$x = 0.454 \cos 30^\circ = 0.393, \quad y = 0.454 \sin 30^\circ = 0.227, \quad z = 0.10 + 0.277 = 0.377 \text{ [m]}.$$

逆運動学の見通し

逆に (x, y, z) が与えられたときは、分解を逆にたどります。

1. 真上から見れば $\theta_1 = \text{atan2}(y, x)$ (図 1.4(b))。

2. 平面内の目標 $\rho = \sqrt{x^2 + y^2}$, $\zeta = z - d_1$ を作る。
3. (ρ, ζ) に対する 2 リンクの逆運動学 (第 1 節) で θ_2, θ_3 を解く。

完全な式は第 5.3 節で扱いますが、本質はすでに第 1 節で終わっています。

実務との接続

- **垂直多関節は世界標準**：産業用ロボットの主流は 6 自由度の垂直多関節で、その根元 3 軸がまさに本節の構造です。慣習的に「最初の 3 軸で手先のだいたいの位置を決め、残り 3 軸（手首）で向きを決める」という役割分担がなされます（第 4・5 章）。
- 教育用の 6 軸アームキット（シリアルバスサーボを 6 個直列につないだタイプ）も同じ構成です。本節の式 (1.8) に自分のアームの寸法 d_1, a_2, a_3 を入れれば、手首の付け根の位置を今日から計算できます。

1.4 本章のまとめ

この章のまとめ

- 順運動学（角度 → 位置）は式 (1.1) のように陽に書け、答えは常に 1 つ。
- 逆運動学（位置 → 角度）は、余弦定理で θ_2 、atan2 の差で θ_1 を求める。解は 0 個・1 個・2 個がありうる。
- 角度の復元には必ず atan2 を使う。
- 脚ロボットも基準軸の取り方が違うだけで同じ式。膝の向きの規約で解を 1 つに絞れる。
- 垂直型 3 自由度は「 θ_1 で回る鉛直平面＋その中の 2 リンク」に分解できる。

第2章 姿勢の記述

この章で学ぶこと

- 「向き（姿勢）」を数学でどう表すかを知る：回転行列・ロール／ピッチ／ヨー・ZYZ オイラー角・単位クォータニオン。
- 回転行列で点の座標変換と回転の合成ができるようになる。
- 4つの表現の長所・短所と、実務での使い分けを説明できるようになる。

第1章では手先の「位置」だけを扱いましたが、コップの水をこぼさず運ぶ、ネジをまっすぐ差し込む、といった作業では手先の「向き」＝**姿勢**の指定が欠かせません。姿勢の表現は1種類ではなく、用途ごとに使い分けします。本章ではその代表4つを学びます。

2.1 姿勢記述のための回転行列

空間に基準の座標系 Σ_A （軸 x_A, y_A, z_A ）を取り、物体に固定した座標系 Σ_B を考えます。物体の姿勢は「 Σ_B の3本の軸が Σ_A から見てどちらを向いているか」で決まります。そこで、 Σ_B の各軸の単位ベクトルを Σ_A の成分で書いて横に並べた 3×3 行列

$${}^A R_B = \begin{bmatrix} \mathbf{x}_B & \mathbf{y}_B & \mathbf{z}_B \end{bmatrix} \quad (2.1)$$

を**回転行列**と呼びます。「列を見れば相手の軸の向きがわかる」のが回転行列のいちばん便利なところです。

座標軸まわりの基本回転は次の3つです（角度 θ は右ねじの向きが正）。

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}, \quad R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.2)$$

ポイント：回転行列の性質

回転行列 R は次を満たします。

- 各列は長さ1で互いに直交する（正規直交）： $R^T R = I$ 。
- よって**逆行列は転置**： $R^{-1} = R^T$ 。逆回転の計算がタダ同然です。
- $\det R = +1$ （鏡映は含まない）。
- 9成分ありますが、上の拘束のため独立な自由度は3つだけです。

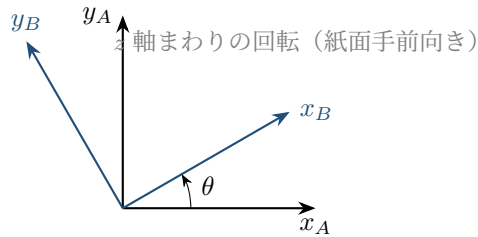


図 2.1 z 軸まわりに θ 回した座標系。回転行列 $R_z(\theta)$ の第 1 列・第 2 列が新しい軸 x_B, y_B の向きになっている。

2.2 回転行列による座標変換

同じ点 P を、 Σ_B で測った座標 ${}^B p$ と Σ_A で測った座標 ${}^A p$ で表すと（両系の原点は一致とする）、

$${}^A p = {}^A R_B {}^B p \quad (2.3)$$

が成り立ちます。回転行列は「 B 系の座標を A 系の座標に翻訳する機械」です。

例題 2.1 点の座標変換

Σ_B は Σ_A を z 軸まわりに 90° 回した座標系である。 Σ_B で $(1, 0, 0)$ にある点は、 Σ_A ではどこか。

解答

$${}^A p = R_z(90^\circ) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

B 系の x 軸方向は、 A 系から見ると y 軸方向——図 2.1 のとおりです。

回転の合成と順序

回転は行列の積で合成できますが、**積の順序で結果が変わります**（非可換）。たとえば $R_z(90^\circ)R_x(90^\circ) \neq R_x(90^\circ)R_z(90^\circ)$ です。覚え方は次の 1 行です。

ポイント：固定軸なら左から、自分の軸なら右から

今の姿勢が R のとき、

- 基準系（固定軸）まわりに回す $\Rightarrow$ 左から掛けて $R' = R_{\text{new}} R$
- 物体系（現在の自分の軸）まわりに回す $\Rightarrow$ 右から掛けて $R' = R R_{\text{new}}$

どちらの軸まわりの回転かを常に意識してください。姿勢バグの大半はここで生まれます。

なお、式 (2.3) の「座標の翻訳」と、「点そのものを回す」操作は同じ行列で書けます（見方が違うだけ）。本書では「翻訳」の見方を基本にします。

2.3 ロール・ピッチ・ヨー角

回転行列は計算には便利ですが、9 個も数字があって人間には読みにくい。そこで「3 つの角度で姿勢を言う」方法が使われます。その代表が**ロール・ピッチ・ヨー角**（RPY 角）です。

基準系の固定軸まわりに、 x 軸まわりロール $\psi \rightarrow y$ 軸まわりピッチ $\phi \rightarrow z$ 軸まわりヨー φ の順で回すと定義すると、前節のルール（固定軸＝左から）より

$$R_{\text{RPY}}(\varphi, \phi, \psi) = R_z(\varphi) R_y(\phi) R_x(\psi) \quad (2.4)$$

です。展開すると

$$R_{\text{RPY}} = \begin{bmatrix} c_\varphi c_\phi & c_\varphi s_\phi s_\psi - s_\varphi c_\psi & c_\varphi s_\phi c_\psi + s_\varphi s_\psi \\ s_\varphi c_\phi & s_\varphi s_\phi s_\psi + c_\varphi c_\psi & s_\varphi s_\phi c_\psi - c_\varphi s_\psi \\ -s_\phi & c_\phi s_\psi & c_\phi c_\psi \end{bmatrix} \quad (2.5)$$

となります（ $c_\varphi = \cos \varphi$ などの略記）。逆に、回転行列 $R = [r_{ij}]$ から RPY 角を取り出すには（ $\cos \phi \neq 0$ のとき）

$$\phi = \text{atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right), \quad \varphi = \text{atan2}(r_{21}, r_{11}), \quad \psi = \text{atan2}(r_{32}, r_{33}) \quad (2.6)$$

とします。

ジンバルロック

ピッチが $\phi = \pm 90^\circ$ になると $\cos \phi = 0$ となり、上の取り出し式が破綻します。このときロールとヨーの回転軸が同じ向きに重なってしまい、**2 つの角度の区別がつかなくなる**のです。これを**ジンバルロック**と呼びます。姿勢そのものが表せなくなるわけではありませんが、(i) 角度の値が不定・不連続になる、(ii) RPY 角のまま補間すると近傍で暴れる、という実害があります。

実務との接続

- **URDF**：ROS で使うロボット記述フォーマット URDF では、リンク間の取り付け姿勢を `rpj="..."` 属性で書きます。本節の定義そのものです。
- **ドローン・IMU**：機体姿勢の表示はロール・ピッチ・ヨーが標準です。IMU（慣性計測ユニット）の出力にもよく使われますが、宙返りなどピッチ $\pm 90^\circ$ をまたぐ機体では、内部処理はクォータニオン（第 2.5 節）で行い、表示だけ RPY にするのが定石です。
- RPY 角は「人が読み書きする場面」に強く、「計算・補間する場面」に弱い、と覚えてください。

2.4 ZYZ-オイラー角

3 角度による表現には軸の選び方の流儀がいくつもあり、ロボットアームでは **ZYZ オイラー角** もよく使われます。**物体系（そのつどの自分の軸）** まわりに $z \rightarrow y \rightarrow z$ の順で α, β, γ 回す定義で、前節のルール（自分の軸＝右から）より

$$R_{\text{ZYZ}}(\alpha, \beta, \gamma) = R_z(\alpha) R_y(\beta) R_z(\gamma). \quad (2.7)$$

回転行列からの取り出しは ($\sin \beta > 0$ の枝で)

$$\beta = \text{atan2}\left(\sqrt{r_{13}^2 + r_{23}^2}, r_{33}\right), \quad \alpha = \text{atan2}(r_{23}, r_{13}), \quad \gamma = \text{atan2}(r_{32}, -r_{31}) \quad (2.8)$$

です。 $\beta = 0$ （または π ）で α と γ の軸が重なり、RPY と同種の特異（不定）が起きます。

ZYZ が好まれる理由は、6 軸ロボットの**手首 3 軸の構造（回す・振る・回す）**とちょうど対応するためです（第 4 章）。手首の関節角と ZYZ 角がほぼ素直に結びつくので、手首の逆運動学が見通しよく書けます。

2.5 単位クォータニオン

準備：軸と角度で回す

どんな姿勢も、ある 1 本の軸 $\mathbf{k}$ （単位ベクトル）まわりに角度 θ だけ回す操作 1 回で作れます（オイラーの定理）。この軸角度表現から回転行列を作る公式が**ロドリゲスの式**です。

$$\text{Rot}(\mathbf{k}, \theta) = \cos \theta I + (1 - \cos \theta) \mathbf{k} \mathbf{k}^T + \sin \theta [\mathbf{k}]_{\times}, \quad [\mathbf{k}]_{\times} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}. \quad (2.9)$$

クォータニオンの定義

軸角度 $(\mathbf{k}, \theta)$ を、次の 4 つ組にパックしたものが**単位クォータニオン**です。

$$\mathbf{q} = (q_w, q_x, q_y, q_z) = \left(\cos \frac{\theta}{2}, \mathbf{k} \sin \frac{\theta}{2} \right), \quad q_w^2 + q_x^2 + q_y^2 + q_z^2 = 1. \quad (2.10)$$

スカラー部 q_w とベクトル部 (q_x, q_y, q_z) からなり、長さが 1 に正規化されています。半角 $\theta/2$ が入るのがミソで、このおかげで回転の合成がクォータニオンの積

$$\mathbf{q}_1 \otimes \mathbf{q}_2 = (w_1 w_2 - \mathbf{v}_1 \cdot \mathbf{v}_2, w_1 \mathbf{v}_2 + w_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2) \quad (2.11)$$

で書けます ($\mathbf{q}_i = (w_i, \mathbf{v}_i)$)。順序の意味は行列の積と同じです)。共役 $\bar{\mathbf{q}} = (q_w, -q_x, -q_y, -q_z)$ が逆回転を表し、点 $\mathbf{p}$ の回転は $(0, \mathbf{p}') = \mathbf{q} \otimes (0, \mathbf{p}) \otimes \bar{\mathbf{q}}$ で計算できます。回転行列への変換は

$$R(\mathbf{q}) = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & 1 - 2(q_x^2 + q_y^2) \end{bmatrix} \quad (2.12)$$

です。

例題 2.2 クォータニオンを作る

z 軸まわり 90° の回転をクォータニオンで表し、点 $(1, 0, 0)$ を回してみよ。

解答 $\mathbf{k} = (0, 0, 1)$, $\theta = 90^\circ$ なので式 (2.10) より

$$\mathbf{q} = (\cos 45^\circ, 0, 0, \sin 45^\circ) = (0.707, 0, 0, 0.707).$$

式 (2.12) に入れると $R(\mathbf{q})$ の第 1 列は $(1 - 2 \cdot 0.5, 2 \cdot 0.5, 0) = (0, 1, 0)$ 。よって $(1, 0, 0)$ は $(0, 1, 0)$ に移ります。例題 2.1 と一致しました。

ポイント：クォータニオンの長所と注意

- **特異なし**：ジンバルロックが原理的に起きません。どんな姿勢も連続に表せます。
- **補間に強い**：2つの姿勢を滑らかにつなぐ計算が簡単です（第 3 章の球面線形補間）。
- **符号の二重性**： $\mathbf{q}$ と $-\mathbf{q}$ は同じ姿勢を表します。比較や補間の前に符号をそろえます。
- **正規化**：数値誤差で長さが 1 からずれるので、演算を重ねたら $\mathbf{q}/\|\mathbf{q}\|$ で正規化し直します。

実務との接続

- **ROS**：座標変換ライブラリ `tf` や `geometry_msgs` の姿勢は、すべてクォータニオンでやり取りされます。姿勢推定 (SLAM・IMU フュージョン) やゲームエンジンの内部表現も同様です。
- **成分の並び順に注意**：本書は (w, x, y, z) の順で書きましたが、ROS や SciPy は (x, y, z, w) の順です。ライブラリ間でデータを渡すときの並び順ミスは、実務で本当によくある事故です。境界では必ずドキュメントを確認してください。

2.6 本章のまとめ

この章のまとめ

- 姿勢の基本表現は回転行列。列＝相手の軸の向き、逆行列＝転置、合成＝積（固定軸は左から、自分の軸は右から）。
- RPY 角は人が読み書きしやすいが、ピッチ $\pm 90^\circ$ でジンバルロック。
- ZYZ オイラー角は 6 軸ロボットの手首構造と相性がよいが、 $\beta = 0$ で特異。
- 単位クォータニオンは特異がなく補間に強い。符号の二重性と正規化、成分の並び順に注意。

表現	成分数	長所	弱点	主な使い所
回転行列	9	変換・合成がそのまま計算	冗長・補間に不向き	座標変換の計算
RPY 角	3	直感的で読み書きしやすい	ジンバルロック	表示・設定ファイル
ZYZ 角	3	手首 3 軸と対応	$\beta = 0$ で特異	アーム手首の解析
クォータニオン	4	特異なし・補間に強い	直感しにくい	通信・補間・姿勢推定

第3章 目標軌道の生成

この章で学ぶこと

- 「経路の形」と「時間の割り付け」を分けて設計する考え方を身につける。
- 作業座標系での計画と関節空間での計画の違いと使い分けを理解する。
- 多項式（3次・5次）と速度台形則で、滑らかな時刻歴を作れるようになる。
- 姿勢の補間は成分ごとにやってはいけない理由と、正しい方法（軸角度・slerp）を知る。

第1章・第2章で「目標の一点」を関節角に翻訳できるようになりました。しかし実機は一瞬でその角度には行きません。今の状態から目標まで、**時間をかけて滑らかにつなぐ道筋——目標軌道——**が必要です。

ポイント：経路と時刻歴を分けて考える

軌道の設計は、

- **経路（path）**：どんな形の道を通るか（直線？円弧？関節空間の直線？）
- **時刻歴（timing）**：その道の上を時間とともにどう進むか（ゆっくり出てゆっくり止まるなど）

の2段階に分けると整理できます。3.1～3.2節が主に経路の話、3.3～3.4節が時刻歴の話です。

実務との接続：PTPとCP、MOVJとMOVL

産業用ロボットの世界では、**PTP制御**（Point To Point：途中経路は問わず目標点へ）と**CP制御**（Continuous Path：手先が指定の経路をたどる）という用語が使われます。ティーチングの命令でいえば、関節補間のMOVJがPTP（≈3.2節）、直線補間のMOVLがCP（≈3.1節の考え方）に対応します。本章はつまり「あのボタンの中身」の章です。

3.1 作業座標系での軌道計画（脚ロボット）

脚ロボットが一步進むとき、宙に浮いている脚（遊脚）の足先は「今の接地点から次の接地点まで、地面を擦らず山なりに」動く必要があります。足先がどんな形を描くかが大事なので、これは**作業座標系で経路を決めるべき**典型例です。

たとえば歩幅 S 、持ち上げ高さ h の一步を、進行率 s ($0 \rightarrow 1$) を使って

$$x(s) = x_0 + Ss, \quad z_{\text{足先高さ}}(s) = h \sin(\pi s) \quad (3.1)$$

と設計します (z は地面からの高さ)。 $s = 0, 1$ で高さ 0 (接地)、 $s = 1/2$ で最高点 h になります。

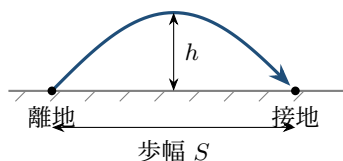


図 3.1 遊脚の足先経路の例。作業座標系で「形」を決める。

実行の手順はこうです。制御周期を Δt (たとえば 10 ms) として、

1. 時刻 t での進行率 $s(t)$ を決める (時間の割り付け。3.3 節の多項式を使うとよい)。
2. 式 (3.1) で足先の目標位置 (x, z) を計算する。
3. 第 1 章 2 節の逆運動学で (θ_1, θ_2) に翻訳する。
4. 関節角をモータへ指令する。 Δt 後に 1 へ戻る。

「毎周期、位置を決めて IK を解く」——これが作業座標系での軌道追従の基本形で、第 5 章の実践にそのままつながります。

実務との接続

式 (3.1) で $s = 0, 1$ の近くの傾きに注目してください。 $s(t)$ を 3.3 節の 3 次多項式にしておくと接地の瞬間の足先速度が 0 に近づき、**着地の衝撃**が減ります。歩行ロボットで「軌道の設計＝乗り心地と壊れにくさの設計」になっている好例です。

3.2 関節空間での軌道計画 (平面 2 自由度ロボット)

途中の経路の形はどのように、 「今の姿勢から目標の姿勢へ移ればよい」 だけなら、もっと簡単な方法があります。関節角そのものを補間するのです。開始角 θ_s 、目標角 θ_g として

$$\theta(s) = \theta_s + s(\theta_g - \theta_s), \quad s: 0 \rightarrow 1. \quad (3.2)$$

これが**関節空間での軌道計画**です。長所は明快です。

- 途中で逆運動学を一度も解かなくてよい (計算が軽く、解なし・複数解の心配もない)。
- 各関節が単調に動くので、可動範囲の管理がしやすい。

一方で重要な性質があります。**関節空間の直線は、手先の直線ではありません。**

具体例で見ましょう。 $l_1 = 0.30$, $l_2 = 0.25$ m の 2 リンクで、 $\theta_s = (0^\circ, 90^\circ)$ から $\theta_g = (90^\circ, 90^\circ)$ へ式 (3.2) で動かします。手先は $(0.300, 0.250)$ から $(-0.250, 0.300)$ へ移りますが、中間 $s = 1/2$ ($\theta = (45^\circ, 90^\circ)$) での手先は $(0.035, 0.389)$ です。2 点を結ぶ直線の中点 $(0.025, 0.275)$ から大きく上へ膨らんだ弧を描くことがわかります。

したがって、「部品をつかんで別の場所へ置く」ような移動はこの方法で十分ですが、「直線に沿って溝を彫る」「まっすぐ引き抜く」ような作業には使えません。そのときは3.1節の方式（作業座標で経路を決めて毎周期IK）に切り替えます。

3.3 時間の多項式に基づく軌道生成

ここからは時刻歴、つまり $s(t)$ や $\theta(t)$ の形を決めます。1つの関節を、時刻0で角度 θ_s （静止）から、時刻 T で角度 θ_g （静止）へ動かすとしましょう。条件は4つあります。

$$\theta(0) = \theta_s, \quad \theta(T) = \theta_g, \quad \dot{\theta}(0) = 0, \quad \dot{\theta}(T) = 0.$$

未知数4つ = **3次多項式**でちょうど決まります。 $\theta(t) = a_0 + a_1t + a_2t^2 + a_3t^3$ において条件を入れると、 $\Delta\theta = \theta_g - \theta_s$ として

$$a_0 = \theta_s, \quad a_1 = 0, \quad a_2 = \frac{3\Delta\theta}{T^2}, \quad a_3 = -\frac{2\Delta\theta}{T^3} \quad (3.3)$$

が得られます。無次元時間 $s = t/T$ を使えば $\theta = \theta_s + \Delta\theta(3s^2 - 2s^3)$ と書いて覚えやすい形です。速度は $t = T/2$ で最大となり、

$$\dot{\theta}_{\max} = \frac{3\Delta\theta}{2T} \quad (3.4)$$

です（平均速度 $\Delta\theta/T$ の1.5倍）。この式で「移動時間 T をどこまで縮められるか」をモータの最高速度から見積もれます。

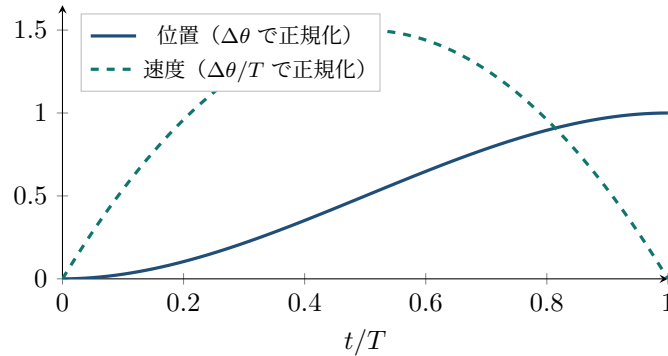


図 3.2 3次多項式軌道。位置はS字、速度は山なりで、両端で速度0になる。

例題 3.1 3次多項式軌道を設計する

関節を 0° から 90° まで $T = 2\text{ s}$ で動かす3次多項式軌道を求め、最大速度を計算せよ。

解答 式 (3.3) より $a_2 = 3 \cdot 90/2^2 = 67.5$, $a_3 = -2 \cdot 90/2^3 = -22.5$ なので

$$\theta(t) = 67.5t^2 - 22.5t^3 \text{ [}^\circ\text{]} \quad (0 \leq t \leq 2).$$

最大速度は式 (3.4) より $\dot{\theta}_{\max} = 3 \cdot 90 / (2 \cdot 2) = 67.5^\circ/\text{s}$ ($t = 1\text{s}$ で発生)。使うモータの最高速度がこれを上回っているか確認します。

3 次多項式は両端の**加速度**が 0 になりません（出だしでトルクが跳ねます）。加速度も両端で 0 にしたいときは、条件 6 つ = **5 次多項式** を使い

$$\theta(t) = \theta_s + \Delta\theta (10s^3 - 15s^4 + 6s^5), \quad s = t/T \quad (3.5)$$

とします。滑らかさが 1 段上がり、振動が問題になる機械で好まれます。

3.4 速度台形則に基づく軌道生成

実務でもっとも広く使われる時刻歴が**速度台形則**です。「最大加速度 A で加速 → 最高速度 V で等速 → A で減速」という、速度グラフが台形になる走らせ方です（図 3.3）。

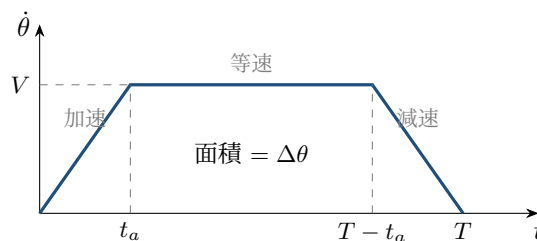


図 3.3 速度台形則。グラフの下の面積が移動量 $\Delta\theta$ になる。

移動量を $\Delta\theta$ (> 0 とする) とすると、

$$t_a = \frac{V}{A} \text{ (加減速時間)}, \quad T = \frac{\Delta\theta}{V} + \frac{V}{A} \text{ (全移動時間)} \quad (3.6)$$

です。等速区間が存在するのは $\Delta\theta \geq V^2/A$ のとき。移動量が小さく $\Delta\theta < V^2/A$ のときは最高速度に達する前に減速に入り、速度グラフは**三角形**になります。このとき

$$\dot{\theta}_{\text{peak}} = \sqrt{A \Delta\theta}, \quad T = 2\sqrt{\Delta\theta/A} \quad (3.7)$$

です。多項式との違いは、**モータの限界値 (V と A) から直接設計**できること。「この距離なら最短何秒か」がすぐ出するため、装置のサイクルタイム設計と相性が抜群です。

例題 3.2 台形則の設計

$\Delta\theta = 90^\circ$, $V = 60^\circ/\text{s}$, $A = 120^\circ/\text{s}^2$ のとき、移動の内訳と所要時間を求めよ。

解答 まず $V^2/A = 3600/120 = 30^\circ \leq 90^\circ$ なので台形になります。式 (3.6) より $t_a = 60/120 = 0.5\text{s}$ 、 $T = 90/60 + 0.5 = 2.0\text{s}$ 。内訳は、加速で $\frac{1}{2}At_a^2 = 15^\circ$ 、等速 1.0s で 60° 、減速で 15° 、合計 90° です。

実務との接続

- **モータドライバの中身**: ステッピングモータのドライバ IC や、シリアルバスサーボの「目標位置+移動時間/速度指定」モードは、内部でこの台形（またはその角を丸めた **S 字** / **ジャーク制限プロファイル**）を生成しています。パラメータ表の Acc / Vel はまさに A と V です。
- 台形則は加速度が階段状に切り替わるため、剛性の低い機械では切り替わりの瞬間に振動が出ます。現場で「S 字にしたら静かになった」と言われるのはこの角を丸めた効果です。

3.5 手先の姿勢に関する軌道計画

位置の補間は各成分を式 (3.2) のように混ぜればよいのですが、**姿勢では同じ手が使えません**。

- 回転行列の 9 成分を成分ごとに補間すると、中間の行列は回転行列でなくなります（列の直交性が壊れる）。
- RPY 角の成分補間は一見動きますが、ジンバルロック付近で暴れ、また「最短の回し方」になりません。

正しい考え方は**相対回転を軸角度に直して、その角度だけを補間**することです。開始姿勢 R_s から目標姿勢 R_g へ動くとき、

1. 相対回転 $R_{\text{rel}} = R_s^T R_g$ を計算する（「 R_s から見て R_g はどれだけ回った先か」）。
2. R_{rel} を軸角度 $(\mathbf{k}, \theta_{\text{tot}})$ に直す：

$$\theta_{\text{tot}} = \arccos \frac{\text{tr } R_{\text{rel}} - 1}{2}, \quad \mathbf{k} = \frac{1}{2 \sin \theta_{\text{tot}}} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}. \quad (3.8)$$

3. 進行率 $s(t)$ (3.3~3.4 節で設計) を使って

$$R(t) = R_s \text{Rot}(\mathbf{k}, s(t) \theta_{\text{tot}}) \quad (3.9)$$

とする（Rot はロドリゲスの式 (2.9)）。

式 (3.9) は「1 本の固定軸まわりに、角度だけを滑らかに増やす」動きで、途中の姿勢はすべて正しい回転行列になり、しかも最短の回し方です。

3.6 単位クォータニオンを使った大円補間

前節と同じことを、クォータニオンなら 1 本の式で書けます。単位クォータニオンは 4 次元単位球面上の点なので、2 つの姿勢 q_0, q_1 を結ぶ自然な道は**球面上の大円**です。これを等速でたどるの

が球面線形補間 (slerp; spherical linear interpolation) で、

$$\mathbf{q}(s) = \frac{\sin((1-s)\Omega)}{\sin \Omega} \mathbf{q}_0 + \frac{\sin(s\Omega)}{\sin \Omega} \mathbf{q}_1, \quad \cos \Omega = \mathbf{q}_0 \cdot \mathbf{q}_1 \quad (3.10)$$

です ($s: 0 \rightarrow 1$ 、 $\mathbf{q}_0 \cdot \mathbf{q}_1$ は 4 成分の内積)。実装上の注意が 2 つあります。

- $\mathbf{q}_0 \cdot \mathbf{q}_1 < 0$ のときは $\mathbf{q}_1 \rightarrow -\mathbf{q}_1$ と符号を反転してから使う (同じ姿勢の 2 表現のうち近い方を選ぶ。忘れると遠回りに回転します)。
- $\Omega \approx 0$ (ほぼ同じ姿勢) では $\sin \Omega$ での割り算が不安定なので、通常の線形補間 + 正規化に切り替える。

slerp は前節の「固定軸まわりの等速回転」と同じ運動を与えます。つまり式 (3.9) のクォータニオン版です。

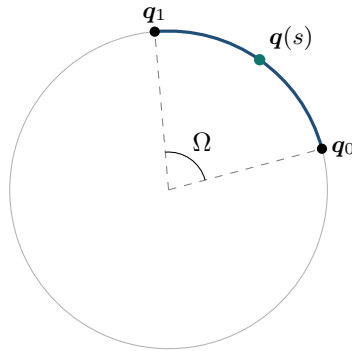


図 3.4 slerp の概念図。単位球面上の大円を等速でたどる (実際は 4 次元球面)。

実務との接続

- 溶接・塗装・バリ取りのように工具の向きが仕上がりを左右する作業では、位置の直線補間と姿勢の slerp を組み合わせて手先軌道を作ります。多くのロボット言語の直線補間命令は、内部で姿勢をこの方式で補間しています。
- CG・ゲームのカメラワークやキャラクタの関節補間も slerp が標準です。ライブラリでは SciPy の Slerp、three.js の Quaternion.slerp など、名前ごと定着しています。

3.7 本章のまとめ

この章のまとめ

- 軌道設計は「経路の形」と「時間の割り付け」に分けて考える。
- 手先の通り道が大事なら作業座標系で計画して毎周期 IK (CP・MOVL 型)。移動できればよいなら関節空間の補間 (PTP・MOVJ 型)。関節空間の直線は手先の直線ではない。

- 静止→静止の滑らかな時刻歴は 3 次多項式 (式 (3.3))。加速度も両端 0 にしたければ 5 次。最大速度は $1.5\Delta\theta/T$ 。
- 速度台形則はモータの限界 V, A から直接設計でき、移動量が小さいと三角形になる。
- 姿勢の補間は成分ごとにやらない。相対回転の軸角度化 (式 (3.9)) か、クォータニオンの slerp (式 (3.10)) を使う。

第4章 運動学の一般的表現

この章で学ぶこと

- 位置と姿勢をひとまとめに扱う同次変換行列を使えるようになる。
- リンクごとに座標系を貼り、少数のパラメータ（リンクパラメータ）で機構を記述する方法を学ぶ。
- 「行列を順に掛けるだけ」でどんな直列リンクの順運動学も書ける、という一般論に到達する。

第1章では機構ごとに図を描いて式を導きました。3自由度まではそれで足りませんが、6自由度になると図だけでは破綻します。本章では、どんな直列リンク機構にも通用する機械的な手続きを整えます。

4.1 リンク座標系と同次変換

座標系 Σ_A から見た座標系 Σ_B の関係は、原点位置 ${}^A\mathbf{p}_B$ （3成分）と姿勢 ${}^A\mathbf{R}_B$ （回転行列）の組で決まります。これを1つの 4×4 行列

$${}^AT_B = \begin{bmatrix} {}^A\mathbf{R}_B & {}^A\mathbf{p}_B \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad (4.1)$$

にまとめたものを同次変換行列と呼びます。点の座標に1を付け足した $\tilde{\mathbf{p}} = (x, y, z, 1)^\top$ を使うと、回転と平行移動が1回の積で書けます。

$${}^A\tilde{\mathbf{p}} = {}^AT_B {}^B\tilde{\mathbf{p}}. \quad (4.2)$$

最大の御利益は連鎖則です。座標系が数珠つなぎになっていれば

$${}^AT_C = {}^AT_B {}^BT_C \quad (4.3)$$

と、間の変換を順に掛けるだけで端から端までの関係が出ます。逆変換も簡単で、

$$T^{-1} = \begin{bmatrix} R^\top & -R^\top \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad (4.4)$$

です（ 4×4 の逆行列計算は不要）。

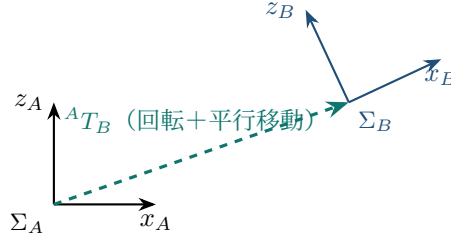


図 4.1 同次変換行列は 2 つの座標系の位置関係（平行移動と回転）をひとまとめに表す。

ポイント：ロボットの運動学＝同次変換の掛け算

ロボットの各リンクに座標系を 1 つずつ貼り、隣どうしの変換 ${}^{i-1}T_i$ （関節角 θ_i の関数）を用意すれば、根元から手先までは連鎖則 (4.3) で

$${}^0T_n = {}^0T_1 {}^1T_2 \cdots {}^{n-1}T_n$$

と書けます。順運動学は「行列を n 回掛ける」作業に統一されるのです。残る問題は「座標系をどう貼り、 ${}^{i-1}T_i$ をどう書くか」だけで、それが次節です。

4.2 リンク座標系の定義とリンクパラメータ

座標系の貼り方を全員が好き勝手にやると式がバラバラになるため、標準の流儀があります。広く使われるのが **DH 法**（Denavit–Hartenberg 法）で、規約は 2 つだけです。

- z_i 軸は関節 $i+1$ の回転軸に取る。
- x_i 軸は隣り合う z 軸どうしの共通垂線の向きに取る。

すると、隣接する座標系の関係は次の 4 つのリンクパラメータだけで書けます。

記号	名前	意味
a_i	リンク長	2 本の関節軸の間の距離（共通垂線の長さ）
α_i	ねじれ角	2 本の関節軸のなす角
d_i	リンクオフセット	共通垂線の足の、軸方向のずれ
θ_i	関節角	軸まわりの回転角（回転関節ではこれだけが変数）

このとき、隣接座標系間の同次変換は「 z まわりに θ_i 回す $\rightarrow z$ 方向に d_i 進む $\rightarrow x$ 方向に a_i 進む $\rightarrow x$ まわりに α_i 回す」の合成で、

$${}^{i-1}T_i = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.5)$$

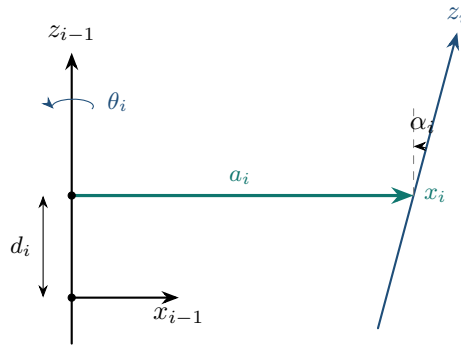


図 4.2 リンクパラメータの模式図。2 本の関節軸 (z_{i-1} と z_i) の位置関係を $a_i, \alpha_i, d_i, \theta_i$ の 4 つで表す。

と 1 つの決まった形になります。機構の情報はパラメータ表に、計算は式 (4.5) に、と完全に分業できるのが DH 法の威力です。

実務との接続

- カタログに DH 表が載っている：産業用ロボットの取扱説明書や研究論文では、機構を図ではなく DH パラメータ表で共有します。表を読むこと＝機構図を読むこと、です。
- URDF との関係：ROS の URDF は、DH と違い各関節の取り付けを「平行移動 xyz + 回転 rpy」で自由に書く方式です（第 2 章の内容がそのまま登場）。DH は手計算・解析向き、URDF は実装・ツール向きで、相互に変換できます。
- DH には「標準 DH」と「修正 DH」という 2 つの流儀があり、添字の付き方が微妙に違います。文献間で式を持ち運ぶときは、どちらの流儀か必ず確認してください（本書は標準 DH）。

4.3 垂直型 3 自由度ロボットの順運動学

第 1.3 節の垂直型 3 自由度ロボットを、DH 法で書き直してみます。腰（鉛直軸）→肩（水平軸）と軸の向きが 90° ねじれているので $\alpha_1 = 90^\circ$ 、肩と肘は平行なので $\alpha_2 = 0$ です。

i	θ_i	d_i	a_i	α_i
1	θ_1	d_1	0	90°
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0

例題 4.1 DH 表から順運動学を再現する

上の表と式 (4.5) から手先位置を計算し、第 1 章の式 (1.8) と一致することを確認めよ。

解答 表の各行を式 (4.5) に代入します。

$${}^0T_1 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^1T_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^2T_3 = \begin{bmatrix} c_3 & -s_3 & 0 & a_3c_3 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

まず平面内の 2 つを掛けると、位置部分は 1T_3 の第 4 列 $= (a_2c_2 + a_3c_{23}, a_2s_2 + a_3s_{23}, 0, 1)^T$ となります (2 リンクの式の再現)。これに 0T_1 を掛けると、位置は

$$\mathbf{p} = \begin{bmatrix} c_1(a_2c_2 + a_3c_{23}) \\ s_1(a_2c_2 + a_3c_{23}) \\ d_1 + a_2s_2 + a_3s_{23} \end{bmatrix}$$

となり、式 (1.8) と完全に一致します。しかも DH 法では、位置と同時に手先の**姿勢** (0T_3 の回転部分) まで自動的に得られています。

4.4 垂直型 6 自由度ロボットの順運動学

実用のアームは、位置 3 自由度 + 姿勢 3 自由度の **6 自由度** が基本です。標準的な構成では、3 自由度の腕の先に、**3 つの回転軸が 1 点で交わる手首 (球面手首)** を付けます。手首の 3 軸は「回す (z)・振る (y)・回す (z)」の並びで、第 2 章の ZYZ オイラー角と対応します。DH パラメータ表の例を示します (d_4 は前腕の長さ、 d_6 は手首中心からフランジまでの長さ)。

i	θ_i	d_i	a_i	α_i
1	θ_1	d_1	0	90°
2	θ_2	0	a_2	0
3	θ_3	0	0	90°
4	θ_4	d_4	0	-90°
5	θ_5	0	0	90°
6	θ_6	d_6	0	0

順運動学は、もはや新しい理屈は何もなく

$${}^0T_6(\theta_1, \dots, \theta_6) = {}^0T_1 {}^1T_2 {}^2T_3 {}^3T_4 {}^4T_5 {}^5T_6 \quad (4.6)$$

を計算するだけです。手で展開すると長大になりますが、プログラムなら数行です (付録 A)。

後の逆運動学で重要になる関係を 1 つだけ導いておきます。手首 3 軸の交点 (**手首中心**) を $\mathbf{p}_w$

とすると、手先（フランジ）位置 $\mathbf{p}$ と姿勢 $R = {}^0R_6$ から

$$\mathbf{p}_w = \mathbf{p} - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (4.7)$$

で求めます（手先の z 軸方向に d_6 だけ戻す）。 $\mathbf{p}_w$ は $\theta_4, \theta_5, \theta_6$ を動かしても動かない——手首中心の位置は根元 3 軸だけで決まる——という事実が、6 自由度の逆運動学を「位置の 3 軸 + 姿勢の 3 軸」に分割する鍵になります（第 5 章）。

実務との接続

6 自由度が標準なのは、空間の剛体の自由度が「位置 3 + 姿勢 3 = 6」だからです。任意の位置・任意の向きに手先を置くには最低 6 軸が要ります。最近増えている 7 軸ロボットは、この 6 に「肘の逃がし」の冗長自由度を 1 つ足したものです。教育用の 6 軸アームキットも同じ思想で、根元 3 軸 + 手首 3 軸の役割分担がそのまま観察できます。

4.5 本章のまとめ

この章のまとめ

- 同次変換行列（式 (4.1)）は位置と姿勢をひとまとめにし、連鎖則 ${}^AT_C = {}^AT_B {}^BT_C$ で数珠つなぎにできる。
- DH 法では z 軸 = 関節軸、 x 軸 = 共通垂線と決め、機構を 4 パラメータ $(a_i, \alpha_i, d_i, \theta_i)$ の表で表す。隣接変換は決まった形 (4.5)。
- どんな直列リンクの順運動学も「表を作って行列を順に掛ける」に統一される。位置だけでなく姿勢も同時に得られる。
- 6 自由度 = 根元 3 軸（位置） + 球面手首 3 軸（姿勢）。手首中心は式 (4.7) で求め、根元 3 軸だけで決まる。

第5章 実践・位置制御と逆運動学

この章で学ぶこと

- 「位置制御されたモータ」で動くロボットの構造を知り、これまでの理論が実機のどこに入るかを理解する。
- 手先を1方向にまっすぐ動かす、という基本タスクを手順に落とせるようになる。
- 垂直型ロボットの位置の逆運動学を完全な式で書き、複数解の扱い方を学ぶ。

最終章では、これまでの道具を実機の枠組みに組み付けます。鍵になるのは「実際のロボットの多くは、関節角度の指令値を受け取って追従するモータで動いている」という事実です。

5.1 位置制御されたモータによって駆動されるロボット

位置制御されたモータとは、図5.1のように、目標角度 θ^d を与えると、内蔵のセンサ（エンコーダ）と制御器（PID 制御など）が勝手に働いて実際の角度 θ を目標へ追従させてくれるモータのことです。使う側から見れば「角度を書き込めばそこへ行ってくれる部品」であり、ラジコン用のサーボモータ、ホビーロボットで使うシリアルバスサーボ、産業用のサーボアンプ+ AC サーボモータは、規模こそ違えみな同じ構造をしています。

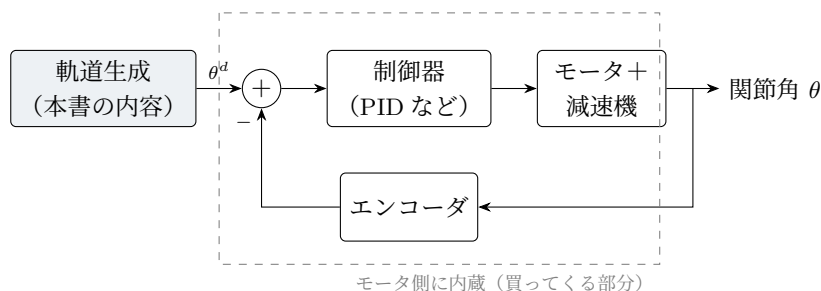


図 5.1 位置制御されたモータの構造。破線内は製品に内蔵されており、私たちの仕事は θ^d を作ること。

この枠組みでは、ロボットを動かす私たちの仕事は明確です。

各関節の目標角度の時系列 $\theta^d(t)$ を作り、一定周期（たとえば 10 ~ 20 ms）でモータへ送り続けること。

そして $\theta^d(t)$ の作り方こそ、第1章（逆運動学）～第3章（軌道生成）で学んだ内容そのものです。

ポイント：目標値をいきなり飛ばさない

位置制御モータは、目標角を階段状にポンと変えても「動いてくれます」。しかし目標と現在の差が大きいと、制御器は全力（最大トルク・最大速度）で追いつこうとし、機体が跳ねたり、過電流で停止したり、ギアを傷めたりします。**必ず第 3 章の軌道（多項式や台形則）で目標値を滑らかに変化させてから送ること。**これが位置制御ロボットを扱ういちばん大事な作法です。

実務との接続

- **シリアルバスサーボ**：Dynamixel や Feetech STS 系のように、1 本の配線に数珠つなぎして ID 指定で角度を書き込むサーボは、教育・研究用アームの定番です（オープンソースの 6 軸アームキット、たとえば SO-ARM101 系もこの構成）。「レジスタに目標位置を書く → 内蔵制御器が追従」という動作は、図 5.1 そのものです。
- **産業用ロボット**：コントローラ内部で軌道生成を行い、各軸のサーボアンプへ周期的に位置指令を配る、という同じ構造をより高速・高精度にしたものです。階層は同じなので、ホビー機で身につけた感覚はそのまま通用します。

5.2 垂直型 6 自由度ロボット：手先一軸の制御

実践の最初の一步として、次のタスクを考えます。

「6 自由度ロボットの手先を、**姿勢を保ったまま**、基準座標系の z 軸に沿って 5cm まっすぐ下ろす。」

部品の挿入や積み下ろしで頻出の動きです。単純に見えますが、**たった 1 方向の直線運動でも、6 つの関節すべてが協調して動く必要があります**（どれか 1 軸を回すだけでは手先は円弧を描いてしまう）。手順は第 3 章の作業座標系計画の応用です。

1. 現在の関節角から順運動学（式 (4.6)）で現在の手先位置 $\mathbf{p}_s$ と姿勢 R_s を求める。
2. 目標を $\mathbf{p}_g = \mathbf{p}_s + (0, 0, -0.05)$, $R_g = R_s$ と置く。
3. 移動時間 T を決め、進行率 $s(t)$ を 3 次多項式（式 (3.3)）で設計する。
4. 制御周期ごとに $\mathbf{p}(t) = \mathbf{p}_s + s(t)(\mathbf{p}_g - \mathbf{p}_s)$, $R(t) = R_s$ を計算し、**逆運動学**で 6 つの関節角に翻訳して送信する。

姿勢も同時に変えたいときは、 $R(t)$ を第 3 章の slerp（式 (3.10)）で補間するだけで、手順は変わりません。

ポイント：解の枝をそろえる

逆運動学には複数の解があるため（次節）、毎周期の IK で**前回と違う枝の解**を選んでしまうと、手先はほぼ同じ場所なのに関節が大きく跳ぶ、という危険な現象が起きます。実装では「前周期の関節角にもっとも近い解を選ぶ」ことで枝を固定します。また、経路の途中が作業領域の外に出ていないか（IK が解を持つか）を事前に確認しておきます。

実務との接続

産業用ロボットのティーチペンダントには、各関節を個別に回す**各軸ジョグ**と、手先を基準座標系の $X/Y/Z$ 方向へまっすぐ動かす**直交ジョグ**があります。直交ジョグのボタンを押している間、コントローラは本節の手順（微小移動→IK→送信）を高速に回し続けています。また、直交ジョグ中に突然アラームで止まることがあります。多くは経路が作業領域の境界や**特異姿勢**（腕が伸びきる等、IK が不安定になる姿勢）に差ししかかったためで、本節の「解が持てるか」の確認がコントローラ内で働いた結果です。

5.3 大まかな場所を実現するアーム部

6 自由度の逆運動学は一見手強いですが、第 4 章の球面手首の性質が効きます。手首中心 p_w は根元 3 軸だけで決まる（式 (4.7)）ので、

手順 1：目標の位置・姿勢から p_w を計算し、**根元 3 軸** $\theta_1, \theta_2, \theta_3$ で p_w を実現する（＝手先の「大まかな場所」を決める）。

手順 2：残った**手首 3 軸** $\theta_4, \theta_5, \theta_6$ で目標姿勢との差を埋める（ZYZ オイラー角の取り出しに帰着）。

と 2 段に分割できます。本節では手順 1、すなわち**アーム部（根元 3 軸）の位置逆運動学**を完成させます。

第 1.3 節の分解に従い、 $p_w = (x_w, y_w, z_w)$ に対して

$$\theta_1 = \text{atan2}(y_w, x_w), \quad (5.1)$$

$$\begin{aligned} \rho &= \sqrt{x_w^2 + y_w^2}, \quad \zeta = z_w - d_1, \\ \cos \theta_3 &= \frac{\rho^2 + \zeta^2 - a_2^2 - a_3^2}{2a_2a_3}, \quad \sin \theta_3 = \pm \sqrt{1 - \cos^2 \theta_3}, \quad \theta_3 = \text{atan2}(\sin \theta_3, \cos \theta_3), \end{aligned} \quad (5.2)$$

$$\theta_2 = \text{atan2}(\zeta, \rho) - \text{atan2}(a_3 \sin \theta_3, a_2 + a_3 \cos \theta_3). \quad (5.3)$$

構造は第 1 章の 2 リンクとまったく同じです。

例題 5.1 位置逆運動学で往復を閉じる

$d_1 = 0.10, a_2 = 0.30, a_3 = 0.25$ m のロボットで、手首中心を $p_w = (0.393, 0.227, 0.377)$ m に置きたい。関節角を求めよ（肘は $\sin \theta_3 < 0$ 側とする）。

解答 式 (5.1) : $\theta_1 = \text{atan2}(0.227, 0.393) = 30.0^\circ$ 。 $\rho = \sqrt{0.393^2 + 0.227^2} = 0.454, \zeta = 0.377 - 0.10 = 0.277$ 。式 (5.2) :

$$\cos \theta_3 = \frac{0.454^2 + 0.277^2 - 0.30^2 - 0.25^2}{2 \cdot 0.30 \cdot 0.25} = 0.868, \quad \theta_3 = -29.8^\circ \quad (\sin \theta_3 = -0.497).$$

式 (5.3) :

$$\theta_2 = \text{atan2}(0.277, 0.454) - \text{atan2}(0.25 \times (-0.497), 0.30 + 0.25 \times 0.868) = 31.4^\circ - (-13.6^\circ) = 45.0^\circ.$$

得られた $(\theta_1, \theta_2, \theta_3) = (30^\circ, 45^\circ, -30^\circ)$ は、例題 1.4 で順運動学に入れた角度そのものです。FK と IK の往復がぴったり閉じました（丸め誤差の範囲）。

複数解とその選び方

6 自由度ロボット全体では、一般に次の組み合わせで最大 $2 \times 2 \times 2 = 8$ 通りの解が現れます。

- 前手系／後手系：式 (5.1) の θ_1 に π を足し、腕を背中側に回しても同じ p_w に届く。
- 肘上げ／肘下げ：式 (5.2) の $\pm$ 。
- 手首の反転：手首 3 軸にも 2 通りの解がある。

どれを選ぶかの実務的な基準は、(1) 前回の解に近いものを選ぶ（動きの連続性、前節）、(2) 関節可動範囲に入っているものを選ぶ、(3) 障害物や特異姿勢を避けられるものを選ぶ、の 3 つです。実際にはリミットで候補が絞られ、8 通り全部が有効なことはまれです。

実務との接続

産業用コントローラでは、この解の枝を形態（コンフィギュレーション）フラグとして明示的に管理します（前／後、肘上／肘下、手首の反転有無など）。教示データにはこのフラグが一緒に保存され、再生時に「同じ枝」で解くことで、教えたときと同じ格好で動くことが保証されます。自作の IK でも同じ発想——解の枝を状態として持ち回る——が安全な実装の基本です。

5.4 本章のまとめ

この章のまとめ

- 多くのロボットは位置制御されたモータの集まり。私たちの仕事は、目標関節角の時系列 $\theta^d(t)$ を作って周期的に送ること。
- 目標値は必ず軌道（多項式・台形則）で滑らかに変化させる。階段状の指令は機体を痛める。
- 手先の直線運動＝「微小移動→IK→送信」の繰り返し。毎周期の IK では前回に近い解を選び、枝の飛び移りを防ぐ。
- 6 自由度の IK は、手首中心（式 (4.7)）で「根元 3 軸の位置問題」と「手首 3 軸の姿勢問題」に分割できる。位置問題の解は式 (5.1)～(5.3)。
- 解は最大 8 通り。連続性・可動範囲・形態フラグで管理する。

付録 A 数値計算で確かめる (Python)

本文の例題は、すべて数行のプログラムで検算できます。「式を写して、数字を入れて、本文と同じ値が出るか確かめる」を習慣にしてください。理解がまるで違ってきます。以下は NumPy / SciPy を使った最小限の例です。

平面 2 自由度の FK と IK (第 1 章)

```
import numpy as np

L1, L2 = 0.30, 0.25

def fk(th1, th2):
    x = L1*np.cos(th1) + L2*np.cos(th1 + th2)
    y = L1*np.sin(th1) + L2*np.sin(th1 + th2)
    return x, y

def ik(x, y, elbow=+1):          # elbow: +1 / -1 = two branches
    c2 = (x*x + y*y - L1*L1 - L2*L2) / (2*L1*L2)
    if abs(c2) > 1.0:
        raise ValueError("target out of workspace")
    s2 = elbow*np.sqrt(1.0 - c2*c2)
    th2 = np.arctan2(s2, c2)
    th1 = np.arctan2(y, x) - np.arctan2(L2*s2, L1 + L2*c2)
    return th1, th2

th = ik(0.40, 0.20, elbow=+1)
print(np.rad2deg(th))           # -> about [-5.5, 71.5] (Example 1.2)
print(fk(*th))                  # -> (0.40, 0.20) round trip OK
```

3 次多項式軌道 (第 3 章)

```
def cubic(th_s, th_g, T, t):
    s = np.clip(t / T, 0.0, 1.0)
    return th_s + (th_g - th_s) * (3*s**2 - 2*s**3)

t = np.linspace(0, 2, 201)
theta = cubic(0.0, np.deg2rad(90), 2.0, t) # Example 3.1
```

slerp による姿勢補間 (第 3 章)

```

from scipy.spatial.transform import Rotation, Slerp

key_rots = Rotation.from_euler("xyz", [[0, 0, 0], [90, 0, 90]], degrees=True)
slerp = Slerp([0.0, 1.0], key_rots)
q_mid = slerp(0.5)
print(q_mid.as_quat())    # note: SciPy order is (x, y, z, w)

```

同次変換で 6 軸の FK（第 4 章）

```

def dh(theta, d, a, alpha):
    ct, st = np.cos(theta), np.sin(theta)
    ca, sa = np.cos(alpha), np.sin(alpha)
    return np.array([[ct, -st*ca, st*sa, a*ct],
                     [st, ct*ca, -ct*sa, a*st],
                     [0.0, sa, ca, d],
                     [0.0, 0.0, 0.0, 1.0]])

# DH table of Sec. 4.4 (d1, a2, d4, d6 are your robot's dimensions)
def fk6(q, d1=0.10, a2=0.30, d4=0.25, d6=0.08):
    params = [(q[0], d1, 0.0, np.pi/2),
              (q[1], 0.0, a2, 0.0),
              (q[2], 0.0, 0.0, np.pi/2),
              (q[3], d4, 0.0, -np.pi/2),
              (q[4], 0.0, 0.0, np.pi/2),
              (q[5], d6, 0.0, 0.0)]

    T = np.eye(4)
    for p in params:
        T = T @ dh(*p)
    return T    # 4x4: rotation + position

T = fk6(np.deg2rad([30, 45, -30, 0, 60, 0]))
print(T[:3, 3])    # hand position

```

参考文献・さらに学ぶために

- 細田耕『実践ロボット制御——基礎から動力学まで』オーム社．本書第 I 部の章立てはこの本の構成に沿っています．動力学（第 II 部相当）へ進むときの本命です．
- 吉川恒夫『ロボット制御基礎論』コロナ社．運動学から力制御までを厳密に扱う日本語の定番書です．
- J. J. Craig, *Introduction to Robotics: Mechanics and Control* (邦訳あり)．DH 法（修正 DH）の代表的教科書です．
- K. M. Lynch and F. C. Park, *Modern Robotics*, Cambridge University Press．著者サイトで PDF と講義動画が無償公開されており、独習に向きます．