

ロボットシミュレーション（１）：順運動学，逆運動学

機械情報工学科 山本 江

2022 年 10 月 14 日

1 はじめに

1.1 機構解析のための計算

計算機によるリンク機構の運動学・動力学計算は機構設計・CG アニメーションなどで不可欠なツールとなっている．動力学計算とはリンクに働く力やモーメントを扱う計算，運動学計算とは位置・速度・加速度に関する（力やモーメントを扱わない）計算をいい，それぞれ順方向と逆方向の計算がある．各計算の入出力と応用例を表 1 に示す．ここで q はリンク系の状態を一意に表す一般化座標 (generalized coordinates)， τ は q に対応する一般化力 (generalized force)， p, R は絶対座標系で表したリンクの位置・姿勢をそれぞれ表す．一般化座標としては関節角を並べたベクトルを用いるのが普通である．このとき対応する一般化力は関節トルクとなる．

表 1: 運動学・動力学計算一覧

計算	入力	出力	応用例	難易度
順運動学 (FK: Forward kinematics)	q	p, R	シミュレーション	易
逆運動学 (IK: Inverse kinematics)	p, R	q	動作計画	やや難
順動力学 (FD: Forward dynamics)	τ, q, \dot{q}	\ddot{q}	シミュレーション	難
逆動力学 (ID: Inverse dynamics)	q, \dot{q}, \ddot{q}	τ	制御	易
[参考] 静力学 (statics)	q	τ	静的バランス	易

1.2 演習の内容

この演習ではロボットシミュレーションの原理を理解するために，上記の各種計算を実装する．演習課題は各回に配布する資料に記載されており，これらを進めて行くことでシミュレータを完成していく形としている．本演習は「MATLAB による二輪ロボット制御」と連動して進行し，最終的に二輪倒立振子の動力学シミュレーションおよび実機実験を行う．

各回の予定は以下の通りである．

第 1 回 基本的なプログラムの原理を習得し，順運動学計算・逆運動学計算を MATLAB 上で実装する．

第 2 回 MATLAB 上で動力学計算の実装を行う．

2 MATLAB の基礎

数値解析プログラム MATLAB MATLAB は米 MathWorks 社¹が販売する数値計算プログラムである。行列の取り扱いを得意としているほか、制御系の設計、画像処理、組込みプログラム開発をはじめとする各種ツールボックスが用意されている。コマンドラインからのインタラクションや m ファイルと呼ばれるスクリプティング言語での使用のほか Simulink と呼ばれるブロック線図ベースのグラフィカル環境が用意されている。

MATLAB は商用であるが、m ファイルと言語的に互換性が高い GNU Octave²と呼ばれるフリーの環境も開発されている。

行列の生成 MATLAB では変数に行列を代入することで動的に行列が生成される。たとえば、

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \end{bmatrix} \quad (1)$$

という 2×3 の行列を生成し変数 A に代入するにはコンソールもしくは m ファイル中に

```
A = [ 1 2 4; 2 3 5 ] ↵
```

とすればよい。ただし、↵ はエンターキーを押すことを意味している。行列を定義している ([] で囲まれている文中) では、(セミコロン) は改行を意味するので、下記の記述でもよい。

```
A = [ 1 2 4 ↵  
2 3 5 ] ↵
```

代入結果をコンソールに表示したくない場合には行末に、(セミコロン) をつける。行中で % より後ろはコメントとみなされるほか、%{と%} で囲まれた文字がコメントとなる。

行列生成のほかの例としては以下の方法がある。

```
A = zeros(4,3); % 4x3 の零行列の生成  
B = ones(2,5); % 2x5 の 1 から構成される行列  
C = 2:8; % 2 から 8 まで 1 刻みで増加する横ベクトル
```

例題 1:5:20 はどのような行列 (ベクトルを含む) を生成するか調べよ。

行列の要素の取り出し 行列全体を表示するには変数名を入力すればよい。たとえば、コンソールで行列 A の内容を確認する場合には A ↵ と入力すればよい。

行列の一要素を取得したい場合はその要素番号を指定すればよい。

$$A = \begin{bmatrix} 1 & 2 & 4 & 8 & 16 \\ 2 & 3 & 5 & 7 & 9 \\ 4 & 5 & 6 & 7 & 8 \\ 8 & 7 & 6 & 5 & 4 \end{bmatrix}$$

の 2 行 1 列要素を取得したい場合は

¹<http://www.mathworks.com>

²<http://www.gnu.org/software/octave/>

```
A(2,1) ↵
```

とすればよい。MATLAB では、C 言語や Python と異なり、行列・配列のインデックスが 1 から始まるのに注意すること。

部分行列を取り出すには行と列の範囲を指定する。A の 2 行目から 4 行目までと 1 列目から 3 列目から構成される部分行列を取り出したい場合は

```
A(2:4,1:3) ↵
```

とすればよい。

応用としては、1,3,4 行目と 2,3,4 列目から構成される部分行列は、A([1,3,4],2:4) ↵ として取得できる。

行もしくは列全体を取り出したい場合には: (コロン) をつかう。たとえば、A の 3 行目全体を取り出したい場合は

```
A(3,:) ↵
```

となる。

行列のサイズ 部分行列を取り出す際にしばしば必要となる情報として、行列のサイズがある。たとえば、式 (1) の A の行数は 2、列数は 3 であるが、これを MATLAB で知るためには size コマンドがある。size(A) は行数、列数からなる配列を返す。

行数、列数いずれかのみを知りたい場合は

```
size(A,1) % 行数  
size(A,2) % 列数
```

とすればよい。

おまけ 1：特殊な変数 ans コンソールに代入を伴わない計算式を入力してエンターキーを押すと ans = から始まる行に計算結果が表示される。この ans は特殊な変数で、直前の代入を行っていない計算式の結果を保存している。

たとえば、2:7 ↵ と入力すると、以下のように表示される。

```
ans =  
  
     2     3     4     5     6     7
```

この結果の第 1 要素から第 3 要素を変数 D に格納したい場合は

```
D = ans(1:3); ↵
```

とすればよい。この変数は、代入を行わない計算をするたびにその計算結果で上書きされるのであくまで臨時の代入先である要注意されたい。


MATLAB の構造体 MATLAB の変数は、数値で初期化されると数値型 (int か double) になるが、構造体とすることもできる。構造体はメンバ変数を定義することで動的に生成される。

たとえば T という変数を、整数値 id と行列 mat というメンバを持つ構造体としたいときは下記のように変数初期化を行えばよい。

```
T.id = 2;  
T.mat = [1 2; 3 4];
```


コンソールで T  とすると

```
T =  
  
    id: 2  
   mat: [2x2 double]
```

と表示されてどのようなメンバを持つかを知ることができる。それぞれの内容を確認するにはたとえば $T.mat$  とすればよい。


m ファイル m ファイルとは $.m$ の拡張子を持つテキストファイルであり、MATLAB のスクリプトや関数のファイルである。たとえば、`test1.m` というファイルを作成し、次の内容を記述したとする。

```
A = [1 2; 3 4];  
e = eig(A);
```

としたとする。このファイルを実行するには MATLAB のコマンドラインで `test1`  と入力すればよい。このファイルが実行されるとワークスペースには A と e が定義される。

MATLAB の関数 MATLAB の関数は m ファイルに `function` キーワードを用いて記述することで定義される。たとえば引数 a および b を取り、その和と差を返す関数 `add_sub` という名前で定義したいとする。これにはまず `add_sub.m` という m ファイルを作成し、

```
function [x, y] = add_sub(a, b)  
x = a + b;  
y = a - b;
```

と記述すればよい。実行するには、MATLAB のコマンドラインで `[x1, x2] = add_sub(2, 4);`  とすれば $x1$, $x2$ にそれぞれ第一戻り値、第二戻り値が格納される。

作成した関数は他の m ファイルからも呼び出すことができる。ただし、呼び出したい関数の m ファイルが、呼び出す側の m ファイルと同じディレクトリ内にあることが必要である（そうでない場合、関数の呼び出しエラーが出る）。異なるディレクトリに関数の m ファイルを置いている場合には、呼び出す側の m ファイルにおいて `addpath` を使ってそのディレクトリを指定すれば呼び出しが可能になる。

MATLAB を使いこなすヒント 試行錯誤しながらスクリプトを書くと前のワークスペースの値を暗黙的に使うことで思った通りにいかないことがある。 `clear all; close all;` を使うとワークスペースの初期化・figure を閉じることができるため活用するとよい。MATLAB はヘルプが充実している。MATLAB ツールバーの“?” ボタンを押すことでヘルプブラウザが起動するので、調べたいコマンドの使い方が分かる。また、関連するコマンドなども表示されるので、より便利な関数が見つかることもある。

3 リンク機構

ロボットマニピュレータは通常複数の剛体リンク (link) が関節 (joint) で接続された構造になっている。このような構造をリンク機構 (kinematic chain または articulated bodies) という。リンクとリンクをつなぐ関節としては 1 軸回りの回転を許す 1 自由度回転関節や 1 方向の並進移動を許す 1 自由度並進関節が使われることが多い。また、ある 1 点の回りの回転をすべて許す 3 自由度の球面関節や、2 自由度のユニバーサルジョイントなどの多自由度関節もある。ネジは回転と並進の両方を行うが、並進方向の移動量は回転量に依存するので、関節の自由度は 1 である。

関節でつながれたリンクをたどっていったとき、もとのリンクに戻るループがあるものを閉リンク機構 (closed kinematic chain)、ループのないものを開リンク機構 (open kinematic chain) と呼ぶ。開リンク機構のうち、リンクが直列に接続されていて枝分かれのないものをシリアルリンク機構、枝分かれのあるものを木構造リンク系と呼ぶことがある。一般のロボットマニピュレータは開リンク機構のものが多く、環境と接触すると閉リンク機構とみなすことができる構造となる。閉リンク機構の例としては (古いタイプの) パンタグラフ、パラレルマニピュレータなどがある。拘束条件の数が多いため、閉リンク機構の運動学・動力学計算は一般に開リンク機構に比べて難しい。

リンク機構の状態を表すのに必要な最小限の変数を一般化座標 (generalized coordinates) といい、その数を自由度数 (degrees of freedom, DOF) という。開リンク機構ではすべての関節の自由度を足し合わせたものが自由度となり、全関節の変位を並べたものを一般化座標とすることができる。一方、閉リンク機構では関節角が独立ではないので、一般に自由度数は関節の自由度数の和よりも小さく、一般化座標の選び方も単純ではない。

リンク機構では、各リンクが独立に運動しているのではなく、関節による拘束を受ける。リンク機構から 1 つのリンクを選んで“ルートリンク”とし、あるリンクから見てルートリンク側に関節で接続されたリンクを“親 (parent) リンク”、反対側に接続されたリンクを“子 (child) リンク”と呼び、親子関係になぞらえてリンク間の接続関係を表現することが多い。ルートリンクを除き、各リンクは 1 つの親リンクを持つ。子リンクの数は任意である。また、子リンクを持たないリンクを特にエンドリンクと呼ぶ。

4 順運動学計算

順運動学計算とは、一般化座標 (関節角) から各リンクの位置・姿勢を求める計算である。広義 (特に逆動力学計算時) には、一般化座標の速度・加速度からリンクの速度・加速度をそれぞれ求める計算を含むこともある。

4.1 剛体の位置・姿勢

剛体の位置・姿勢は、図 1 のようにその剛体に固定された座標系のある座標系から見た位置・姿勢で定義される。位置は通常座標系の位置ベクトル \mathbf{p} で表す。姿勢は 3×3 の姿勢行列 \mathbf{R} で表すのが直感的にわかりやすい。姿勢行列を $\mathbf{R} = (\mathbf{r}_x \mathbf{r}_y \mathbf{r}_z)$ と分解すると、各列は物理的には座標系の x, y, z 軸の方向を表している。したがって、以下の関係が成り立つ。

$$\mathbf{r}_x^T \mathbf{r}_x = 1, \mathbf{r}_y^T \mathbf{r}_y = 1, \mathbf{r}_z^T \mathbf{r}_z = 1 \quad (2)$$

$$\mathbf{r}_x^T \mathbf{r}_y = 0, \mathbf{r}_y^T \mathbf{r}_z = 0, \mathbf{r}_z^T \mathbf{r}_x = 0 \quad (3)$$

これらの関係を用いると

$$\mathbf{R}^T \mathbf{R} = \mathbf{I} \quad (4)$$

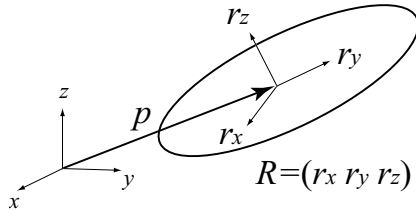


Fig. 1: 剛体リンクの位置と姿勢

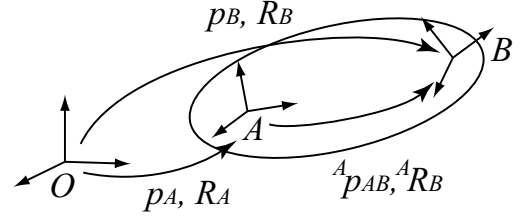


Fig. 2: 絶対座標系における位置・姿勢の計算

(I は単位行列) が導けるので, $R^{-1} = R^T$ という関係があることがわかる.

姿勢を行列 R で表す方法は直感的にわかりやすいが, 回転の 3 自由度を 9 つのパラメータで表しているため各成分が独立でなく, 式 (2), 式 (3) のように複雑な依存関係がある (9 つのパラメータに 6 つの拘束条件があるので, 実質独立なパラメータは 3 つである). 姿勢を最小限のパラメータで表す方法としてロール・ピッチ・ヨー (roll-pitch-yaw) 角, オイラー角 (Euler Angles) などがある. 3 つの独立なパラメータで姿勢を表すことができ, 直感的にも比較的わかりやすいためよく使われているが, いずれも特異点 (3 つのパラメータが一意に決められない姿勢) があって全空間を均一に表現できないという欠点があるため, 利用するときには十分注意する必要がある. 特異点が存在しない表現方法として, 4 つのパラメータを用いる単位クォータニオン, オイラーパラメータが知られている (この両者は実質的に等価である).

4.2 相対位置・姿勢

図 2 のように, 絶対座標系 O において位置 ${}^O p_A$, 姿勢 ${}^O R_A$ である剛体に固定された座標系で表した位置が ${}^A p_{AB}$ の点の絶対座標系における位置 ${}^O p_B$ は次式で計算される.

$${}^O p_B = {}^O p_A + {}^O R_A {}^A p_{AB} \quad (5)$$

また, 座標系 A で ${}^A R_B$ と表される座標系 B は, 絶対座標系では

$${}^O R_B = {}^O R_A {}^A R_B \quad (6)$$

と表される. ${}^A p_{AB}$ や ${}^A R_B$ のように絶対座標系以外の座標系で表された位置・姿勢を, その座標系に対する相対位置・相対姿勢と呼ぶことがある.

以下では, このように左肩の添え字で基準となる座標系を表す. またこの添え字のない位置・姿勢は, 特に断りのない限りは絶対座標系で表されているものとする.

4.3 順運動学計算

順運動学計算に限らず, リンク系の運動計算では親から子へ, あるいは子から親へと順に計算していく再帰計算を行うことが多い. 順運動学計算においては, 関節角から親リンクに対する子リンクの相対位置・姿勢が計算できるので, ベースリンクの位置・姿勢が既知であれば 4.2 節の方法を使ってベースリンクから順に子リンクの位置・姿勢を計算していくことができる. 以下ではベースリンクを 0 として各リンクに一意的な番号が付けられているものとし, リンク j がリンク i の子リンクであるときに, リンク i の位置・姿勢からリンク j の位置・姿勢を求めるアルゴリズムを導出する.

リンク i の絶対座標系で表した位置・姿勢 ${}^0 p_i, {}^0 R_i$ とリンク i 座標系で表したリンク j の位置・姿勢 ${}^i p_{ij}, {}^i R_j$ がわかっているとき, リンク j の絶対座標系で表した位置・姿勢はそれぞれ

$${}^0 p_j = {}^0 p_i + {}^0 R_i {}^i p_{ij} \quad (7)$$

$${}^0 R_j = {}^0 R_i {}^i R_j \quad (8)$$

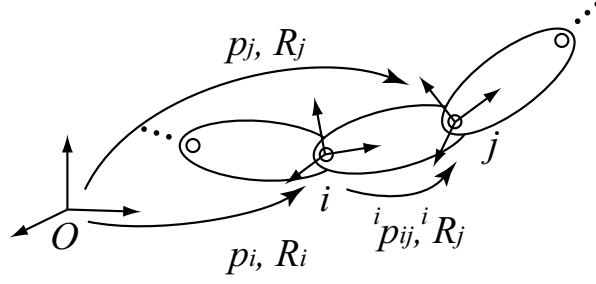


Fig. 3: リンク i の位置・姿勢からリンク j の位置・姿勢を計算

となる (図 3 参照).

相対位置・姿勢の計算方法は関節の種類によって異なるが, ここでは z 軸回りの回転関節について示す. 回転関節は並進移動しないので, 相対位置 ${}^i p_{ij}$ は一定である. 相対姿勢 ${}^i R_j$ は以下のようにして計算する. 関節角が 0 のときのリンク i に対するリンク j の相対姿勢を ${}^i R_{j0}$ とすると, 関節角が q_j のとき ${}^i R_j$ は

$${}^i R_j = {}^i R_{j0} R_z(q_j) \quad (9)$$

で計算される. ここで $R_z(\theta)$ は z 軸回りに角度 θ だけ回転させる変換行列を表し, 以下のように書ける.

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (10)$$

さらに一般的にはロドリゲスの式を用いることで任意の軸 \mathbf{a} 周りの回転行列 \mathbf{R} を得ることができる.

$$\mathbf{R}(\mathbf{a}, \theta) = \begin{cases} \mathbf{I} + \left[\frac{\mathbf{a}}{\|\mathbf{a}\|} \times \right] \sin \theta + \left[\frac{\mathbf{a}}{\|\mathbf{a}\|} \times \right]^2 (1 - \cos \theta) & \|\mathbf{a}\| \neq 0 \\ \mathbf{I} & \|\mathbf{a}\| = 0 \end{cases} \quad (11)$$

ただし, $[\times]$ は歪対称行列を示すものとし, 次式で与えられるものとする.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad [\mathbf{x} \times] = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \quad (12)$$

5 課題 1 : 順運動学シミュレータの実装

MATLAB シミュレータプログラムの入手 まず, 本演習用のプログラムをダウンロードして解凍する.

指定の URL より MATLAB プログラム (class1.zip) をダウンロードして各自の作業ディレクトリに展開する. 解凍パスワードは演習中に掲示する.

この課題では図 4 に示す 7 自由度のマニピュレータの順運動学計算を実装する. 各リンクの座標系は右手系になるように選択する.

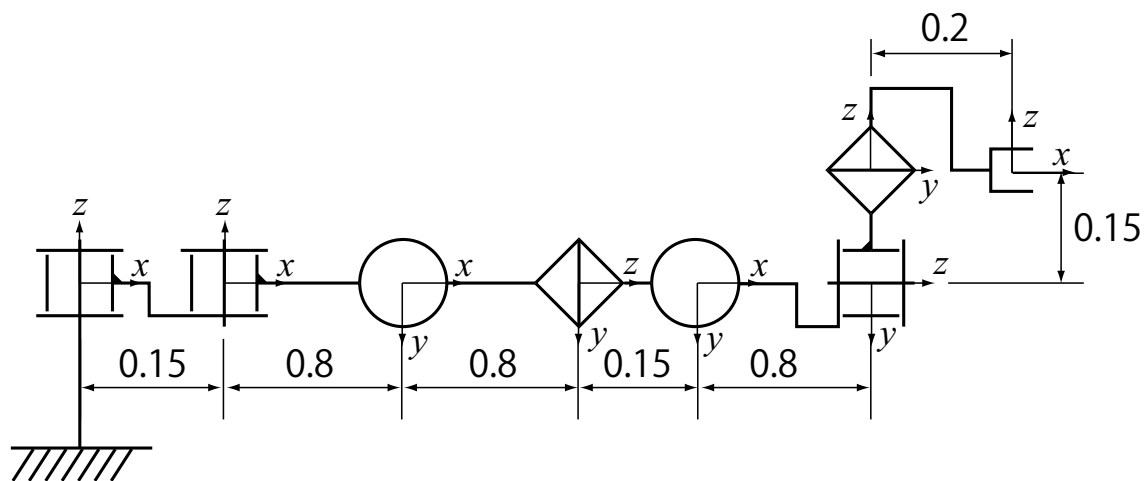


Fig. 4: アームの構造

順運動学シミュレーションの流れ 順運動学は下記の流れで処理を行う (`fk_sample.m` 参照).

1. リンク情報の設定 (`CreateChain.m`)
2. 関節角度の設定 (配列で定義)
3. 順運動学計算 (`ForwardKinematics.m`)
4. (必要に応じて) リンク系の可視化 (`DrawFunction.m`)

項目 2~3 を繰り返すことで関節角の変化に対する振る舞いを計算できる.

リンク情報の設定 まず, 展開したファイルの中に `CreateChain.m` があるのでその中に必要なパラメータ類を設定する. 図 4 を見て, このような構造がどのように実装されるのかを確認せよ.

このファイルでは大きく分けて次の 3 つの処理を行っている.

1. リンク接続情報の設定
2. リンクの幾何学構造の設定
3. リンク姿勢の初期化

リンクは接続情報や幾何学情報などを持つ構造体の配列として定義する. 設定用のメンバ変数のほか, リンク系の状態を示すメンバ変数が存在する.

リンク情報が設定された構造体 (ここでは `chain` としている) を生成するには

```
chain = CreateChain;
```

とすればよい.

順運動学計算の実装 演習課題では次に配列に関節角度を格納し, その配列を `ForwardKinematics.m` に渡して順運動学計算を行う. `ForwardKinematics.m` ではまず `SetJointAngle.m` を呼び出して `chain` 構造体の角度情報を更新する. 次に, ベースリンクから順に配列をたどり, 順運動学計算を行っていく.

表 2: リンク構造体の幾何学構造設定メンバ変数

メンバ名	説明	本文中での記号
linknum	リンクの数	${}^i\mathbf{p}_{ij}$
id	リンクを管理するための ID 番号	
endeffector_id	エンドエフェクタの ID 番号	
mother	親リンクの ID を保持（ベースリンクの場合は 0 を設定）	
child	子リンクの ID を保持（エンドエフェクタの場合は 0 を設定）	
axis	自リンク座標系から見た回転軸の方向	
rel_pos	親リンク座標系から見たリンク座標原点（親リンク座標系で表現）	
init_att	親リンク座標系から見たリンク座標の角度が 0 のときの回転行列（親リンク座標系で表現）	${}^i\mathbf{R}_{j0}$

表 3: リンク構造体の質量特性設定メンバ変数

メンバ名	説明	本文中での記号
rel_cog	自リンク座標系から見たリンク重心位置（自リンク座標系で表現）	${}^j\mathbf{p}_{Cj}^j$
mass	リンクの質量	m_j
I	自リンク座標系から見たリンクの重心周りの慣性モーメント	${}^j\mathbf{I}_j$

表 4: リンク構造体の関節変数関連メンバ変数

メンバ名	説明	本文中での記号
q	関節角度	q_i
qd	関節角速度	\dot{q}_i
qdd	関節角加速度	\ddot{q}_i

表 5: リンク構造体の運動学関連メンバ変数

メンバ名	説明	本文中での記号
pos	絶対座標系表記のリンク位置	${}^0\mathbf{p}_i$
cog_pos	絶対座標系表記のリンク重心位置	${}^0\mathbf{p}_{Ci}$
R	絶対座標系表記のリンク姿勢行列	${}^0\mathbf{R}_i$

表 6: リンク構造体の動力学関連メンバ変数

メンバ名	説明	本文中での記号
vel	絶対座標系表記のリンク速度	${}^0\dot{\mathbf{p}}_i$
ang_vel	絶対座標系表記のリンク角速度	${}^0\dot{\boldsymbol{\omega}}_i$
acc	絶対座標系表記のリンク加速度	${}^0\ddot{\mathbf{p}}_i$
ang_acc	絶対座標系表記のリンク角加速度	${}^0\ddot{\boldsymbol{\omega}}_i$
cog_acc	絶対座標系表記のリンク重心加速度	${}^0\ddot{\mathbf{p}}_{Ci}$
inertia_force	絶対座標系表記のリンクに作用する合力	${}^0\mathbf{F}_j$
inertia_moment	絶対座標系表記のリンクに作用する合モーメント	${}^0\mathbf{N}_j$
force	絶対座標系表記のリンクに作用する外力	${}^0\mathbf{f}_j$
moment	絶対座標系表記のリンクに作用する外モーメント	${}^0\mathbf{n}_j$

課題 1-1

順運動学のコアな計算を実装する.

- 展開したファイルの中にある ForwardKinematics.m の中で順運動学計算のアルゴリズムを実装せよ. 引数の入出力部分などはすでに実装してある. なお, 関節軸ベクトルから回転行列を得るにはロドリゲスの式を用いるとよい (Rodrigues.m).
- fk_sample.m を実行し, ロボットが正しく描画されていることを確認せよ.

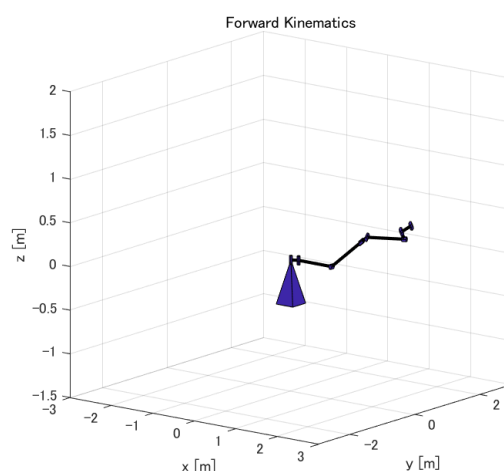


Fig. 5: 課題 1-1 で ForwardKinematics.m を正しく実装した場合の fk_sample.m の実行結果

ロボットの動きの可視化 ロボットの動きを可視化するために, 本演習では DrawFunction という関数を用意した. この関数に chain を渡すことでリンク系の構造を描画できる. すなわち, chain で記述されたロボットを描画するには

```
DrawFunction(chain)
```

とすればよい.

アニメーションをしたい場合には、for ループ内で DrawFunction 関数を呼び出し、pause (MATLAB 組み込み関数) を用いて、描画のための適度な待ち時間を設けたり、毎フレームせずに、間引いて表示するなどするとよい。

計算のループを構成するには for 文を使えばよい。C++での for 文

```
for(int i=1; i<=N; i++)
{
// some process
}
```

に対応する MATLAB の for 文は以下のようになる。

```
for i=1:N
% some process
end
```

課題 1-2

fk_assignment.m というファイルを新たに作成し、以下の順運動学計算プログラムの内容を実装せよ。

- 関節角度が \mathbf{q}_{ini} から \mathbf{q}_{fin} に移動する場合を考え、上記関節角度間を 100 分割して線形補完したものに関するのエンドエフェクタ軌道を計算し、各フレーム毎に DrawFunction 関数でロボットの姿勢を描画せよ。ただし、 \mathbf{q}_{ini} と \mathbf{q}_{fin} はそれぞれ以下のように与える。

$$\mathbf{q}_{ini} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

$$\mathbf{q}_{fin} = \begin{bmatrix} \pi/7 & -\pi/7 & -\pi/7 & \pi/7 & \pi/7 & \pi/7 & \pi/7 \end{bmatrix}^T$$

- いま、 i フレーム目のエンドエフェクタ位置を ${}^0\mathbf{p}_{E,i} = \begin{bmatrix} {}^0x_{E,i} & {}^0y_{E,i} & {}^0z_{E,i} \end{bmatrix}^T$ ，エンドエフェクタの姿勢行列を ${}^0\mathbf{R}_{E,i} = [{}^0r_{nm,E,i}]$ とするとき、計算結果 \mathbf{x}_E を

$$\mathbf{x}_E = \begin{bmatrix} {}^0x_{E,1} & {}^0x_{E,2} & \cdots & {}^0x_{E,n} \\ {}^0y_{E,1} & {}^0y_{E,2} & \cdots & {}^0y_{E,n} \\ {}^0z_{E,1} & {}^0z_{E,2} & \cdots & {}^0z_{E,n} \\ {}^0r_{11,E,1} & {}^0r_{11,E,2} & \cdots & {}^0r_{11,E,n} \\ {}^0r_{12,E,1} & {}^0r_{12,E,2} & \cdots & {}^0r_{12,E,n} \\ {}^0r_{13,E,1} & {}^0r_{13,E,2} & \cdots & {}^0r_{13,E,n} \\ {}^0r_{22,E,1} & {}^0r_{22,E,2} & \cdots & {}^0r_{22,E,n} \\ {}^0r_{23,E,1} & {}^0r_{23,E,2} & \cdots & {}^0r_{23,E,n} \\ {}^0r_{33,E,1} & {}^0r_{33,E,2} & \cdots & {}^0r_{33,E,n} \end{bmatrix} \quad (13)$$

となるように適当な変数に保存せよ。

- load('fk_solution_check.mat') とすることで、正解のエンドエフェクタ軌道計算結果を読み込むことができるので、自分で求めたエンドエフェクタの軌跡 \mathbf{x}_E との誤差ノルムを計算し、実装した計算が正しいことを確認せよ。TA のチェックを受けること。

二つの値を等分割するには linspace を用いればよい。

6 逆運動学問題

ロボットの運動計画やCGにおけるキャラクタの運動生成には、関節角を一つ一つ指定する代わりに手や足の位置を指定することで関節角を計算する逆運動学計算ができれば便利である。しかし、関節値とリンクの位置・姿勢は非線形な関係にあり、その逆関数を求めるのは容易ではない。リンク系の逆運動学の計算方法には大きく分けて以下の2つがある。

1. 解析的解法：幾何学的な関係や、逆三角関数などを用いて順運動学関数の逆関数を陽に求める方法。単純な機構に対しては有効だが、多自由度の機構に適用するのは難しい。
2. 数値的解法：収束計算により近似的に数値解を求める方法。任意の機構に対して適用できるが、高精度の解を得るには長い計算時間がかかる。

本演習では、順運動学で取り上げたアームを対象として数値的解法を実装し、動作確認を行う。

7 逆運動学問題の数値的解法

逆運動学問題の解法として、数値的に近似解を求める方法がいくつかある。これらの方法は、関節速度とリンク速度の関係が瞬間的には線形であることを利用しており、任意の機構に適用できる汎用性を持つ。

あるリンクの位置 \mathbf{p} を関節角 \mathbf{q} から計算する非線形な順運動学関数を $\mathbf{p} = \mathbf{f}(\mathbf{q})$ とおき、時間微分すると

$$\dot{\mathbf{p}} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J}(\mathbf{q}) \dot{\mathbf{q}} \quad (14)$$

という線形な関係が得られる。この係数行列 \mathbf{J} を \mathbf{p} の \mathbf{q} に関するヤコビ行列 (Jacobian matrix) という。 \mathbf{J} は一般に関節値に依存する。

本節ではヤコビ行列を用いた逆運動学の数値解法を2つ示す。

7.1 ヤコビ行列の逆行列による方法

ヤコビ行列 \mathbf{J} の逆行列が存在する場合には、逆行列を用いて式 (14) を関節速度について解くと、リンク速度から関節速度を計算する式が得られる。

$$\dot{\mathbf{q}} = \mathbf{J}^{-1} \dot{\mathbf{p}} \quad (15)$$

この関係を使うと数値的に逆運動学計算のアルゴリズムが導出できる。与えられたリンク位置 \mathbf{p}_d を満たす関節角 \mathbf{q}_d を計算することを考える。適当な関節角 \mathbf{q}_0 を初期値として、以下の手順で収束計算を行う。

1. 初期設定

$$\mathbf{q} = \mathbf{q}_0, \mathbf{p} = \mathbf{f}(\mathbf{q}_0)$$

2. 位置誤差を計算

$$\Delta \mathbf{p} \triangleq \mathbf{p}_d - \mathbf{p}$$

3. 位置誤差の大きさ $|\Delta \mathbf{p}|$ が与えられた許容精度 ϵ 以下であれば $\mathbf{q}_d = \mathbf{q}$ として計算終了。

4. 位置誤差に適当なゲイン k をかけ、リンク位置を $k\Delta \mathbf{p}$ だけ移動させるために必要な関節値の修正量を計算する。

$$\Delta \mathbf{q} = k \mathbf{J}^{-1} \Delta \mathbf{p} \quad (16)$$

5. 関節値とリンク位置を更新し, 2. に戻る.

$$\mathbf{q} = \mathbf{q} + \Delta \mathbf{q}, \mathbf{p} = \mathbf{f}(\mathbf{q})$$

今回考えている問題では, 関節速度 7 次元のベクトルに対して, エンドエフェクタに生じる角度と角速度は 6 次元で表されるので, ヤコビ行列 \mathbf{J} は 6×7 の行列となる. このような冗長マニピュレータのようにヤコビ行列が正方でない場合は, 逆行列の代わりに擬似逆行列を使う方法がある. なお, この場合にはヤコビ行列の擬似逆行列の零空間を用いて冗長自由度を生かすこともできる.

いま, 擬似逆行列を $\#$ で表すとすると, 冗長なリンク系のリンク速度と手先速度の関係は次式で与えられる.

$$\dot{\mathbf{q}} = \mathbf{J}^\# \dot{\mathbf{p}} + (\mathbf{I} - \mathbf{J}^\# \mathbf{J}) \boldsymbol{\xi} \quad (17)$$

ここで, 右辺第一項はこの冗長問題の最小二乗解を与える. 右辺第二項は零空間を表し, $\boldsymbol{\xi} \in R^n$ は任意のベクトルである. ただし n は \mathbf{q} の次元とした.

この零空間は必ず存在するとは限らないが, 零空間に重要度の低いタスクを指定すれば $\dot{\mathbf{p}}$ は厳密に満たしつつ $\boldsymbol{\xi}$ をできるだけ実現するような運動を求めることができる.

7.2 ヤコビ行列の転置による方法

以下のような評価関数を考える.

$$E = \frac{1}{2} (\mathbf{p}_d - \mathbf{p})^T (\mathbf{p}_d - \mathbf{p}) \quad (18)$$

逆運動学計算は, E を最小化する \mathbf{q} を求める計算と考えることができる. すなわち, この方法は \mathbf{p} と \mathbf{p}_d の間にポテンシャル関数 E であらわされる引力が働いている場合に \mathbf{p} が動く軌跡を求めることに等しい.

式 (18) を関節角 \mathbf{q} で偏微分すると次式が得られる.

$$\frac{\partial E}{\partial \mathbf{q}} = (\mathbf{p}_d - \mathbf{p})^T \left(-\frac{\partial \mathbf{p}}{\partial \mathbf{q}} \right) = -(\mathbf{p}_d - \mathbf{p})^T \mathbf{J} \quad (19)$$

評価関数の値を小さくするためには, 式 (16) の代わりに次式により \mathbf{q} を更新すればよいことになる.

$$\Delta \mathbf{q} = -k \left(\frac{\partial E}{\partial \mathbf{q}} \right)^T = k \mathbf{J}^T \Delta \mathbf{p} \quad (20)$$

7.3 ヤコビ行列の汎用計算法 (基礎ヤコビ行列の計算法)

リンク位置 \mathbf{p} の関節値に関するヤコビ行列 $\mathbf{J} = \partial \mathbf{p} / \partial \mathbf{q}$ を計算する方法として, 順運動学計算式を書き下し, 定義通り関節値に関して偏微分するのが最も素直であるが, 計算量が多い上に一般の機構に対応するのは難しい. しかし, 幾何学的な関係を使うと任意の機構に対して非常に簡単にヤコビ行列を計算できる.

簡単のため 1 自由度の回転関節 i を考える. ヤコビ行列からこの回転関節に対応する列 \mathbf{j}_i を取り出すと, \mathbf{j}_i は 6×1 のベクトルになる. このうち, 初めの 3 要素は“関節を $\dot{q}_i = 1(\text{rad/s})$ で回したときのリンク並進速度”を表し, 後の 3 要素は同様にリンク回転速度を表す. よって, 図 6 に示す各量を用いると \mathbf{j}_i は以下のようにして計算できる.

$$\mathbf{j}_i = \begin{pmatrix} \mathbf{a}_i \times (\mathbf{p} - \mathbf{p}_i) \\ \mathbf{a}_i \end{pmatrix} \quad (21)$$

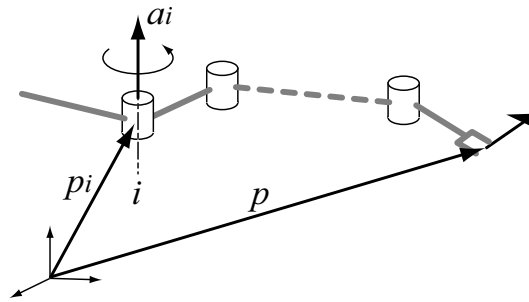


Fig. 6: ヤコビ行列の計算

8 課題2：逆運動学シミュレータの実装

この課題では順運動学で取り上げた図4に示す7自由度のマニピュレータの逆運動学計算を実装する。

逆運動学シミュレーションの流れ 逆運動学計算は以下の流れで行われる。

1. リンクの設定 (CreateChain.m)
2. 関節角度の初期化 (SetJointAngle.m)
3. 目標手先位置・姿勢の設定 (配列で定義)
4. 逆運動学計算 (InverseKinematics.m)
5. (必要に応じて) ロボットの可視化 (DrawFunction.m)

項目3～5を繰り返すことで、目標手先位置・姿勢の軌道に対する関節角軌道を計算することができる。

逆運動学計算 配布したシミュレータでは目標位置・姿勢を配列に格納し、その配列を InverseKinematics.m に渡して逆運動学計算を行う。InverseKinematics.m は目標位置・姿勢に対して 30 回収束計算を行い、30 回以内に誤差が十分収束したら繰り返し計算を終了する。

基礎ヤコビ行列の計算は CalcBasicJacobian() で行える。また、誤差の計算は CalcConfigError() で行える。この結果を用いて収束具合を判断できる。

逆運動学計算の実行

課題 2

ik_assignment.m というファイルを新たに作成し、逆運動学シミュレータを作成せよ。

- 関節角度の初期位置 \mathbf{q}_{ini} , エンドエフェクタの目標位置 \mathbf{p}_d , 目標姿勢 \mathbf{R}_d はそれぞれ以下のように設定する。

$$\mathbf{q}_{ini} = \begin{bmatrix} -\pi & \pi/3 & -\pi/4 & \pi/3 & \pi/7 & 2 * \pi/5 & \pi/7 \end{bmatrix}$$
$$\mathbf{p}_d = \begin{bmatrix} -0.4687 \\ -1.072 \\ 1.063 \end{bmatrix}, \quad \mathbf{R}_d = \begin{bmatrix} 0.6506 & -0.6383 & -0.4114 \\ 0.5977 & 0.7646 & -0.2411 \\ 0.4684 & -0.0890 & 0.8790 \end{bmatrix}$$

- InverseKinematics.m を用いて

- ヤコビ行列の擬似逆行列を用いる方法
- ヤコビ行列の転置を用いる方法

上記目標軌道を実現する関節角度を計算せよ。なお、InverseKinematics.m にはヤコビ行列の擬似逆行列を用いる逆運動学が実装されているのでヤコビ行列の転置を用いる場合には適宜修正せよ。擬似逆行列・転置収束に用いるゲインの値は擬似逆行列・転置で異なるので注意し必要に応じて ik_gain の値を変更すること。

- 上記2つの方法それぞれについて計算結果が目標に収束していることをグラフで図示し、TA のチェックを受けること。

余裕があれば順運動学のときと同様、ik_solution_check_inv.mat（擬似逆行列のときの関節角軌道の正解）と ik_solution_check_trans.mat（転置のときの関節角軌道の正解）を用いて正解と同じ解が得られているかを確認せよ。

9 課題の提出

課題全てについて TA のチェックに合格したものは、作成した以下の.m ファイルを ITC-LMS に提出すること。提出したものは退出して構わない。

- ForwardKinematics.m
- fk_assignment.m
- InverseKinematics.m
- ik_assignment.m