第3部　ロボット力学 (robot dynamics)

宇都宮大学工学研究科 吉田勝俊

2017.2.3 版

目 次

8　オイラー・ラグランジュ方程式

オイラー・ラグランジュ方程式の一般形を，次に示す．

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} + \frac{\partial \mathcal{D}}{\partial \dot{q}_i} = \mathcal{F}_i \quad (i = 1, \cdots, n) \tag{8.1}$$

$\boldsymbol{q} = [q_i]$ はこのあとすぐ述べる**一般化座標**である．その他の諸量を，表 1 にまとめておく．これらの具体形を用意し，(8.1) に代入すると運動方程式が求まる．**一般化力** $\boldsymbol{\mathcal{F}} = [\mathcal{F}_i]$ については **9 節**の段階で述べる．

表 1: オイラー・ラグランジュ方程式の諸量

ラグランジュ関数 $\mathcal{L}$	$\mathcal{L} = \mathcal{T} - \mathcal{U}$　　($\mathcal{T}$：運動エネルギー, $\mathcal{U}$：ポテンシャル)		
運動エネルギー $\mathcal{T}$	並進	$\frac{1}{2}m\lvert\dot{\boldsymbol{x}}\rvert^2$	$\boldsymbol{x}, \theta, \boldsymbol{\omega}, I, J$ は慣性系で測った成分
	回転 (2 次元)	$\frac{1}{2}I\dot{\theta}^2$　　I は慣性モーメント	
	回転 (3 次元)	$\frac{1}{2}\boldsymbol{\omega}\cdot(J\boldsymbol{\omega})$　　$\boldsymbol{\omega}$ は角速度ベクトル J は慣性テンソル	
ポテンシャル $\mathcal{U}$	重力	$m\,g\,h$	
	線形ばね	$\frac{1}{2}k\,x^2$	
散逸関数 $\mathcal{D}$	クーロン摩擦	$\frac{1}{2}\nu R\,\mathrm{sgn}(\dot{x})\dot{x}$	
	粘性抵抗	$\frac{1}{2}c\,\dot{x}^2$	
	慣性抵抗	$\frac{1}{2}D\,\lvert\dot{x}\rvert\dot{x}^2$	
一般化力 $\boldsymbol{\mathcal{F}} = [\mathcal{F}_i]$		$\sum_k \frac{\partial x_k}{\partial q_i} f_k$　　**9 節** p23	

8.1　一般化座標 $\boldsymbol{q} = [q_i]$

例えば，平面上の剛体の姿勢は，位置ベクトル $\boldsymbol{x} = [x_i]$ と回転角 θ で完全に定まる．この剛体の機構学的な自由度は 3 である．そこで，この 3 つの変数を並べた 3 次元ベクトル，

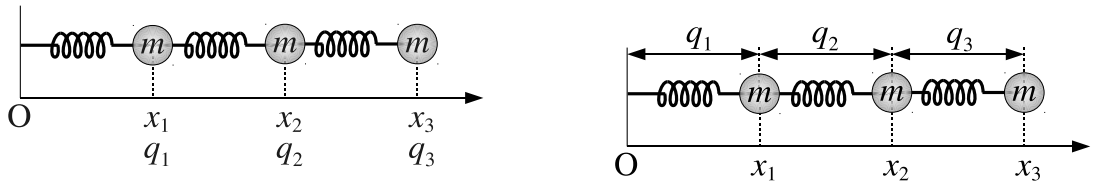
$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} := \begin{bmatrix} x_1 & [\mathrm{m}] \\ x_2 & [\mathrm{m}] \\ \theta & [\mathrm{rad}] \end{bmatrix} \quad \text{とか} \quad \begin{bmatrix} \theta & [\mathrm{rad}] \\ x_1 & [\mathrm{m}] \\ x_2 & [\mathrm{m}] \end{bmatrix}$$

を導入する．通常の空間座標とは異なり，単位の違う量が同列に並んでいる．このように，機構学的な自由度を表す変数を，単位を気にしないで，ただ並べて作った数ベクトル $\boldsymbol{q}$ を，**一般化座標** (generalized coordinate) という．

変数のとり方や並べ順は，各人で好きに決めてよい．例えば，ばねで継がれた 3 質点の水平運動を考えるとき，図 1 のような 2 種類のとり方がありうる [1)]．左は原点 O からの絶対距離を並べた一般化座標，右は相対距離を並べた一般化座標である．いずれも 3 質点の機構学的な自由度を過不足なく表現できているので，どちらを採用してもよい．運動方程式の解として，絶対距離が欲しい人は左を，相対距離が欲しい人は右を採用するだろう．

もちろん，物理的には同じ運動なのだから，運動方程式を解いたあとに相互に変換することも可能である．解析力学の強みは，こうした二度手間を省き，各人のニーズに合せて，欲しい変量を解として持つ運動方程式を導けるところにある．

[1)] もちろん他の取り方もある．考えてみよ！



(a) 絶対距離方式 (慣性系)　　(b) 相対距離方式 (非慣性系)

図 1: 一般化座標の例

8.2 座標変換 — 慣性座標と一般化座標

ニュートンの第 2 法則 $m\ddot{\boldsymbol{x}} = \boldsymbol{f}$ が成立する座標系を**慣性系**と呼んだ．これで測った座標を，**慣性座標**と呼ぼう．図 1(b) の原点 O を慣性系の原点とすると，q_1 は慣性座標だが，q_2, q_3 は慣性座標ではない．これを忘れると，運動エネルギー $\mathcal{T}$ の計算を間違う．

なぜなら，表 1 の諸量のうち，運動エネルギー $\mathcal{T}$ の定義式は，慣性座標で書かれたものだからだ．ゆえに，例えば，図 1(b) の中央の質点の運動エネルギーは，

$$\text{(誤)} \quad \mathcal{T} = \frac{m}{2}\dot{q}_2^2$$
$$\text{(正)} \quad \mathcal{T} = \frac{m}{2}\dot{x}_2^2 = \frac{m}{2}(\dot{q}_1 + \dot{q}_2)^2$$

のように，慣性座標 x_2 から計算しなければならない．

この種の見落しを防ぐためにも，一般化座標 $\boldsymbol{q} = [q_i]$ をとったら，必ず，慣性座標 $\boldsymbol{x} = [x_i]$ への座標変換を書き下しておく．図 1 については，

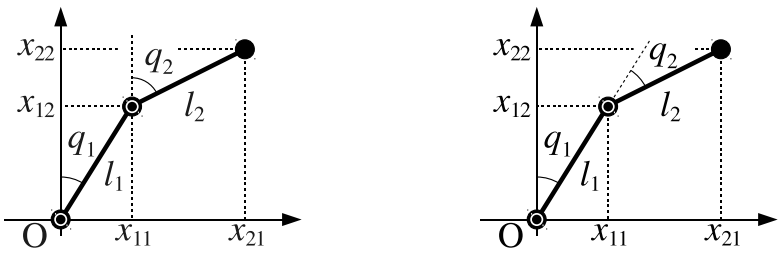
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad \text{※図 1 (a)，恒等変換}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_1 + q_2 \\ q_1 + q_2 + q_3 \end{bmatrix} \quad \text{※図 1 (b)}$$

となる．(a) の場合は，慣性座標 $[x_i]$ をそのまま一般化座標 $[q_i]$ としているが，もちろんこうした取り方でもよい．

それ以外の，$\mathcal{U}$，$\mathcal{D}$，$\mathcal{F}_i$ の定義式は，慣性座標でなくとも使える．ようするに，定義式がニュートンの第 2 法則から導出されたものは，いったん慣性座標に戻して計算する．

例題 3.1　2 節リンクの一般化座標 $\boldsymbol{q} = [q_i]$ を，次のように 2 種類とる．左は空間に対する絶対角によるもの，右は相対角によるものである．



(a) 絶対角方式　　(b) 相対角方式

中間節の慣性座標を $\boldsymbol{x}_1 = (x_{11}, x_{12})^T$，先端の慣性座標を $\boldsymbol{x}_2 = (x_{21}, x_{22})^T$ とする． $\boldsymbol{q}$ から $\boldsymbol{x}_1, \boldsymbol{x}_2$ への座標変換を求めよ．

► **解答例**　絶対角方式の一般化座標に対して，

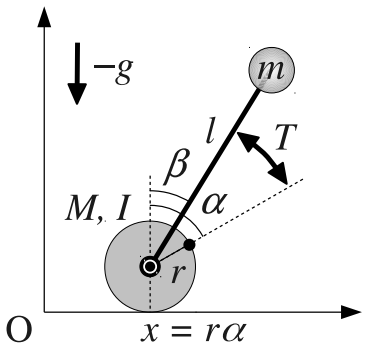
$$\begin{bmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \end{bmatrix} = \begin{bmatrix} l_1 \sin q_1 \\ l_1 \cos q_1 \\ l_1 \sin q_1 + l_2 \sin q_2 \\ l_1 \cos q_1 + l_2 \cos q_2 \end{bmatrix} \quad \text{※ (a) 絶対角方式} \tag{8.2}$$

相対角方式の一般化座標に対して，

$$\begin{bmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \end{bmatrix} = \begin{bmatrix} l_1 \sin q_1 \\ l_1 \cos q_1 \\ l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \\ l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \end{bmatrix} \quad \text{※ (b) 相対角方式} \tag{8.3}$$

となる．

8.3　車輪型倒立ロボット



図の座標系を慣性系とする．一般化座標を (α, β) とすると，各質点の直交座標は，

$$\boldsymbol{x}_M = \begin{bmatrix} x \\ r \end{bmatrix}, \quad \boldsymbol{x}_m = \mathrm{Trans}(\boldsymbol{x}_M)\,\mathrm{Rot}(-\beta) \begin{bmatrix} 0 \\ l \end{bmatrix} = \begin{bmatrix} x + l \sin\beta \\ r + l \cos\beta \end{bmatrix}, \quad x = r\alpha \tag{8.4}$$

となる．

例題 3.2　オイラー・ラグランジュ方程式 (8.1) p18 を利用して運動方程式を導け．

► **解答例**　まず，(8.4) を時間微分すると，速度は，

$$\dot{\boldsymbol{x}}_M = \begin{bmatrix} \dot{x} \\ 0 \end{bmatrix}, \quad \dot{\boldsymbol{x}}_m = \begin{bmatrix} \dot{x} + l\dot{\beta}\cos\beta \\ -l\dot{\beta}\sin\beta \end{bmatrix}, \quad \dot{x} = r\dot{\alpha} \tag{8.5}$$

となり，運動エネルギーは，

$$\mathcal{T} = \frac{M}{2}\dot{\boldsymbol{x}}_M^2 + \frac{m}{2}\dot{\boldsymbol{x}}_m^2 + \frac{I}{2}\dot{\alpha}^2 \tag{8.6}$$

$$= \frac{M}{2}\dot{x}^2 + \frac{m}{2}\left((\dot{x} + l\dot{\beta}\cos\beta)^2 + (-l\dot{\beta}\sin\beta)^2\right) + \frac{I}{2}\dot{\alpha}^2 \tag{8.7}$$

$$= \frac{M}{2}\dot{x}^2 + \frac{m}{2}\left(\dot{x}^2 + 2l\dot{x}\dot{\beta}\cos\beta + l^2\dot{\beta}^2\right) + \frac{I}{2}\dot{\alpha}^2 \tag{8.8}$$

$$= \frac{Mr^2}{2}\dot{\alpha}^2 + \frac{m}{2}\left(r^2\dot{\alpha}^2 + 2lr\dot{\alpha}\dot{\beta}\cos\beta + l^2\dot{\beta}^2\right) + \frac{I}{2}\dot{\alpha}^2 \tag{8.9}$$

$$= \frac{(M+m)r^2 + I}{2}\dot{\alpha}^2 + mlr\dot{\alpha}\dot{\beta}\cos\beta + \frac{ml^2}{2}\dot{\beta}^2 \tag{8.10}$$

となる．ポテンシャルは，

$$\mathcal{U} = mgl\cos\beta + C \quad (C \text{ は定数}), \tag{8.11}$$

となる．これより，この系のラグランジュ関数は，

$$\mathcal{L} = \mathcal{T} - \mathcal{U} = \frac{(M+m)r^2 + I}{2}\dot{\alpha}^2 + mlr\dot{\alpha}\dot{\beta}\cos\beta + \frac{ml^2}{2}\dot{\beta}^2 - mgl\cos\beta - C \tag{8.12}$$

となる．これをオイラー・ラグランジュ方程式 (8.1) p18 に代入すると運動方程式が求まる．
実際，必要な偏微分を計算すると，

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial \dot{\alpha}} = \{(M+m)r^2 + I\}\dot{\alpha} + mlr\dot{\beta}\cos\beta \\
&\qquad \Longrightarrow \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\alpha}} = \{(M+m)r^2 + I\}\ddot{\alpha} + mlr(\ddot{\beta}\cos\beta - \dot{\beta}^2\sin\beta), \quad \frac{\partial \mathcal{L}}{\partial \alpha} = 0, \\
&\frac{\partial \mathcal{L}}{\partial \dot{\beta}} = mlr\dot{\alpha}\cos\beta + ml^2\dot{\beta} \implies \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\beta}} = mlr(\ddot{\alpha}\cos\beta - \dot{\alpha}\dot{\beta}\sin\beta) + ml^2\ddot{\beta}, \\
&\frac{\partial \mathcal{L}}{\partial \beta} = -mlr\dot{\alpha}\dot{\beta}\sin\beta + mgl\sin\beta
\end{aligned}$$

となる．また，関節トルク T が自由度 α と β に発生する一般化力をそれぞれ $\mathcal{F}_\alpha, \mathcal{F}_\beta$ と表記しておく．これらを (8.1) に代入すると運動方程式は，

$$\begin{aligned}
\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\alpha}} - \frac{\partial \mathcal{L}}{\partial \alpha} &= \{(M+m)r^2 + I\}\ddot{\alpha} + mlr(\ddot{\beta}\cos\beta - \dot{\beta}^2\sin\beta) \\
&= \{(M+m)r^2 + I\}\ddot{\alpha} + (mlr\cos\beta)\ddot{\beta} - mlr\dot{\beta}^2\sin\beta = \mathcal{F}_\alpha \qquad (8.13) \\
\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\beta}} - \frac{\partial \mathcal{L}}{\partial \beta} &= mlr(\ddot{\alpha}\cos\beta - \dot{\alpha}\dot{\beta}\sin\beta) + ml^2\ddot{\beta} - (-mlr\dot{\alpha}\dot{\beta}\sin\beta + mgl\sin\beta) \\
&= (mlr\cos\beta)\ddot{\alpha} + ml^2\ddot{\beta} - mgl\sin\beta = \mathcal{F}_\beta \qquad (8.14)
\end{aligned}$$

となる．ベクトル形式で加速度 $\ddot{\alpha}, \ddot{\beta}$ をくくり出すと，運動方程式は，

$$\underbrace{\begin{bmatrix} (M+m)r^2 + I & mlr\cos\beta \\ mlr\cos\beta & ml^2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix}}_{\ddot{\boldsymbol{q}}} = \underbrace{\begin{bmatrix} mlr\dot{\beta}^2\sin\beta \\ mgl\sin\beta \end{bmatrix}}_{\boldsymbol{b}} + \underbrace{\begin{bmatrix} \mathcal{F}_\alpha \\ \mathcal{F}_\beta \end{bmatrix}}_{\boldsymbol{\mathcal{F}}} \tag{8.15}$$

のように整理できる．一般化力 F_α, F_β の具体形は **9 節**で計算する．

8.4 運動方程式の 1 階化

運動方程式は，加速度を含むので，数学的には 2 階の常微分方程式となる．ところが，コンピュータは 2 階微分の処理が苦手なので，運動方程式を 1 階の常微分方程式に見せかけるテクニックが必要になる．これを運動方程式の **1 階化**という．

簡単のため，単振り子の運動方程式，

$$ml^2\ddot{x} = -mgl\sin x \tag{*}$$

を例にとる (x は角度)．この方程式の 2 階微分 $\ddot{x}$ を見かけ上消すため，1 階微分を他の変数に置いた $\dot{x} = y$ を連立する．

$$\begin{cases} \dot{x} = y \\ ml^2\ddot{x} = -mgl\sin x \end{cases}$$

これに，変換式 $\dot{x} = y$ を微分したもの $\ddot{x} = \dot{y}$ を代入すると，

$$\begin{cases} \dot{x} = y \\ \dot{y} = -\frac{g}{l}\sin x \qquad \text{※整理した} \end{cases} \tag{**}$$

となり，見かけ上，2 階微分が消せる．このような式変形，

$$\text{2 階 1 連立 (*)} \overset{\substack{\text{ダミー変数} \\ y=\dot{x}}}{\Longrightarrow} \text{1 階 2 連立 (**)}$$

を，**1 階化**という．1 階化の前 (*) と後 (**) で変数の数こそ違うが，方程式としては等価であり，同じ解を持つ [2)]．

さらに，3 階微分を含む場合 $\dddot{x} = f(\ddot{x}, \dot{x}, x)$ でも，2 階微分までを別の変数 $\dot{x} = y$, $\ddot{x} = z$ に置けば，

$$\begin{cases} \dot{x} = y \\ \dot{y} = z \\ \dot{z} = f(z, y, x) \end{cases} \tag{8.16}$$

のように 1 階化できる．同様の手順で変数を増やせば，一般に，

$$\text{n 階 1 連立} \overset{\text{ダミー変数}}{\Longrightarrow} \text{1 階 n 連立}$$

という 1 階化が可能である．変数が多いときは，通し番号の変数，

$$x_1 = x, \quad x_2 = \dot{x}, \quad x_3 = \ddot{x}, \quad \cdots, \quad x_n = \frac{d^{n-1}x}{dt^{n-1}}$$

を使うのが便利である．

例題 3.3　次の運動方程式を 1 階化せよ．

$$\begin{cases} \ddot{\alpha} = h_1(\alpha, \dot{\alpha}, \beta, \dot{\beta}) \\ \ddot{\beta} = h_2(\alpha, \dot{\alpha}, \beta, \dot{\beta}) \end{cases} \tag{8.17}$$

新しい変数は，$x_1 = \alpha, \quad x_2 = \dot{\alpha}, \quad x_3 = \beta, \quad x_4 = \dot{\beta}$ とせよ．

▶ **解答例**　新しい変数と，その微分 $\dot{x}_1 = \dot{\alpha}, \quad \dot{x}_2 = \ddot{\alpha}, \quad \dot{x}_3 = \dot{\beta}, \quad \dot{x}_4 = \ddot{\beta}$ を使うと，

$$\begin{cases} \dot{x}_1 = x_2 & \because \dot{x}_1 = \dot{\alpha} = x_2 \\ \dot{x}_2 = h_1(x_1, x_2, x_3, x_4) & \\ \dot{x}_3 = x_4 & \because \dot{x}_3 = \dot{\beta} = x_4 \\ \dot{x}_4 = h_2(x_1, x_2, x_3, x_4) & \end{cases} \tag{8.18}$$

という 1 階化が得られる．

実習 3.1　**Code 4** を実行せよ．(8.15) において $\mathcal{F}_\alpha = \mathcal{F}_\beta = 0$ としたときの自由運動がアニメーション表示される．なお，プログラム中の運動方程式は，(8.15) を加速度について

$$\ddot{\boldsymbol{q}} = A^{-1}\Big(\boldsymbol{b} + \boldsymbol{\mathcal{F}}\Big) \quad \left(\equiv \boldsymbol{h} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}\right) \tag{8.19}$$

のように解き，これを例題 3.3 p22 のように **1 階化**した形式で書かれている．

[2)] (*) から (**) が導いたが，(**) から (*) も導ける．ゆえに方程式 (*) と方程式 (**) は，数学的に等価であり，同じ解を持つ．

9　一般化力とその応用

9.1　一般化力 $\boldsymbol{\mathcal{F}} = [\mathcal{F}_i]$

力の単位はニュートン (N) だった．こうした単位を無視して定義される力ベクトル $\boldsymbol{\mathcal{F}} = [\mathcal{F}_i]$ を，**一般化力** (generalized force) という．

例えば，平面上の剛体に作用させられる外力は，力 $\boldsymbol{F} = [F_i]$ [N] とトルク T [N·m] である．これらを，単位を気にせず並べてしまった，

$$\boldsymbol{\mathcal{F}} = [\mathcal{F}_i] = \begin{bmatrix} F_1 & [\mathrm{N}] \\ F_2 & [\mathrm{N}] \\ T & [\mathrm{N{\cdot}m}] \end{bmatrix} \tag{9.1a}$$

を，一般化力という．

9.2　一般化力の作用

一般化力の作用を知るには，対応する一般化座標の運動を想像すればよい．簡単のため，平面上の剛体を考える．その運動法則 $m\ddot{\boldsymbol{X}} = \boldsymbol{F}$, $I\ddot{\theta} = T$ を成分で見ると，

$$\begin{cases} m\ddot{X}_1 = F_1 \\ m\ddot{X}_2 = F_2 \\ I\ddot{\theta} = T \end{cases} \tag{9.1b}$$

である．ここで F_1 の作用とは，対応する X_1 を増減させる作用である．同様に，F_2, T はそれぞれ X_2, θ を増減させる．一般化力も同じである．一般化力の各成分 $\mathcal{F}_i$ は，対応する一般化座標の成分 q_i を増減させる．

例題 3.4　例題 3.1 の一般化座標 $\boldsymbol{q} = (q_1, q_2)$ に対する一般化力を $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2)$ とする．(a) と (b) のそれぞれについて，一般化力の各成分 $\mathcal{F}_i$ が，2 節リンクのどの自由度を増減させるか説明せよ．(b) の $\mathcal{F}_2$ を**関節トルク**という．

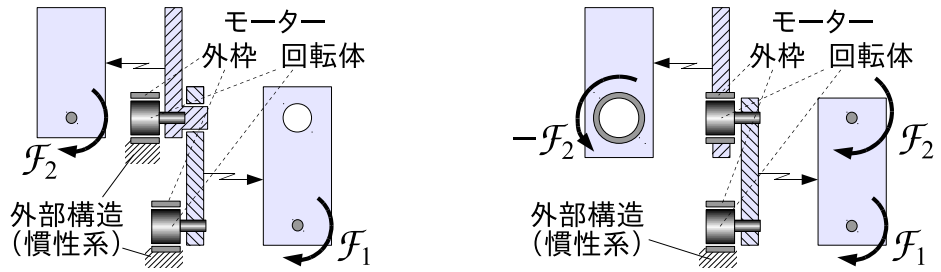
> ▶ **解答例**　$\mathcal{F}_1$ の効果は (a), (b) 共通で，慣性系と第 1 リンクのなす角 q_1 を増減させる．これに対して，(a) の $\mathcal{F}_2$ は第 2 リンクの角度 q_2 だけを増減させ，q_1 には作用しない．(b) の $\mathcal{F}_2$ は第 1 リンクと第 2 リンクの相対角 q_2 を増減させる．

例題 3.5　例題 3.4 の説明を踏まえて，$\mathcal{F}_2$ がリンクに与える作用が，(a) 絶対角方式と (b) 相対角方式でどのように違うか，考察せよ．

> ▶ **解答例**　図 2 のように，(a) 絶対角方式の $\mathcal{F}_2$ は，絶対角 q_2 を直接変化させるトルクなので，その作用の対象は第 2 リンクのみである．これに対して，(b) 相対角方式の $\mathcal{F}_2$ は，相対角 q_2 を押し開く (閉じる) ように働くので，第 1 リンクと 2 リンクに対して，同時に逆向きのトルク $\mathcal{F}_2, -\mathcal{F}_2$ が作用する．

図 2 に示した構造を，もう少し詳しく補足しておこう．(a) 絶対角方式では，モーターの外枠は 2 個とも，外部構造 (慣性系など) に対して回転してはならない．となると上側のモータの設置方法が課題になる．例えば，外部構造にモータの外枠を固定し，回転体 (軸) の回転を回転伝達用ケーブル [3] で伝えるような設計もありうるが，工業的にはあまり見られない構造だと思う．その一方で，(b) 相対角方式では，外部構造に固定するのは，下側

[3] 自転車のブレーキケーブルの高級バージョンみたいなやつ．



(a) 絶対角方式　　(b) 相対角方式

図 2: 2 節リンクにおける一般化力の作用

のモータのみでよい．上側のモータは，外枠を第 1 リンクに固定し，回転軸を第 2 リンクに固定すればよい (逆でもいいが)．(b) 相対角方式のモーターがリンクに発揮するトルクを，**関節トルク**という．この構造はロボットの関節に応用されている．

関節トルクは，内力である．なぜなら，関節トルクは隣接するリンクに同時に逆向きのトルク (例えば，$\mathcal{F}_2$ と $-\mathcal{F}_2$) を発生するから，総和すればキャンセルする．したがって，外力を受けずに浮遊する [4] ロボットの姿勢を，こうした関節トルクで変更しても，重心運動の軌跡は元のまま維持されることになる [5]．

9.3 一般化力の座標変換

以上，一般化力 $\mathcal{F}_i$ の作用を定性的に理解した．では，定量的にはどうだろう．一般化力はどんな値をとるのか．これは，次の算法で計算できる．

算法 3.1 直交座標系における力 $\boldsymbol{F} = [F_i]$ とその着力点 $\boldsymbol{x} = [x_i]$ を考える．着力点の直交座標 $\boldsymbol{x}$ と，他の任意の一般化座標 $\boldsymbol{q} = [q_i]$ との座標変換が，

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{q}) = \begin{bmatrix} x_1(q_1, \cdots, q_m) \\ \vdots \\ x_n(q_1, \cdots, q_m) \end{bmatrix} \tag{9.2}$$

で与えられるとき，$\boldsymbol{q} = [q_i]$ に関する一般化力 $\boldsymbol{\mathcal{F}} = [\mathcal{F}_i]$ は，

$$\mathcal{F}_i = \sum_{k=1}^{n} \frac{\partial x_k}{\partial q_i} F_k = \left[\frac{\partial x_1}{\partial q_i}, \cdots, \frac{\partial x_n}{\partial q_i} \right] \begin{bmatrix} F_1 \\ \vdots \\ F_n \end{bmatrix} =: \left(\frac{\partial \boldsymbol{x}}{\partial q_i} \right)^T \boldsymbol{F} \tag{9.3}$$

で計算できる．表記短縮のため，さらに $\mathcal{F}_i$ をベクトルにまとめると，

$$\boldsymbol{\mathcal{F}} \equiv \begin{bmatrix} \mathcal{F}_1 \\ \vdots \\ \mathcal{F}_m \end{bmatrix} = \begin{bmatrix} \frac{\partial x_1}{\partial q_1} & \cdots & \frac{\partial x_n}{\partial q_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_1}{\partial q_m} & \cdots & \frac{\partial x_n}{\partial q_m} \end{bmatrix} \begin{bmatrix} F_1 \\ \vdots \\ F_n \end{bmatrix} =: \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}} \right)^T \boldsymbol{F} \tag{9.4}$$

という表現を得る．$\partial \boldsymbol{x}/\partial \boldsymbol{q}$ は座標変換 $\boldsymbol{x}(\boldsymbol{q})$ のヤコビ行列である．

[4] 重力はあってよい．

[5] ただし，空気抵抗がなければ．空気抵抗という外力は，ロボットの形状に依存するので，ロボットの姿勢が変れば外力が変化し，ゆえに重心の軌跡も変化する．

驚くべきことに，算法 3.1 の $\boldsymbol{F}$ と $\boldsymbol{x}$ は，実は，直交座標系における力とその着力点でなくてもよい．すなわち，算法 3.1 は，次のように読み替えることができる．

算法 3.2　1 つめの一般化座標 $\boldsymbol{x} = [x_i]$ に関する一般化力を $\boldsymbol{F} = [F_i]$ とし，2 つめの一般化座標 $\boldsymbol{q} = [q_i]$ に関する一般化力を $\boldsymbol{\mathcal{F}} = [\mathcal{F}_i]$ とするとき，両者の関係は，(9.2), (9.3) で計算できる．

算法 3.1〜3.2 の力の換算法は，物理的な直感に自信が持てないときに非常に頼りになる．計算すれば答えが出るからだ．

9.4　車輪型倒立ロボット

8.3 節 p20 の車輪型倒立ロボットを考える．

例題 3.6 (車輪型倒立ロボットの一般化力)　関節トルク T が自由度 α, β におよぼす一般化力 $\boldsymbol{\mathcal{F}}_T \equiv (\mathcal{F}_\alpha, \mathcal{F}_\beta)^T$ を求めよ．

▶ **解答例**　トルク T は自由度 $\theta \equiv \alpha - \beta$ に働く一般化力なので，これを $T = F_\theta$ と表記する．このとき算法 3.1〜3.2 より，

$$F_\alpha = \frac{\partial\theta}{\partial\alpha}F_\theta = 1 \cdot T = T, \quad F_\beta = \frac{\partial\theta}{\partial\beta}F_\theta = -1 \cdot T = -T$$

となる．ゆえに $\boldsymbol{\mathcal{F}}_T = (T, -T)^T$ を得る [6]．

以上の $\boldsymbol{\mathcal{F}}_T$ を，運動方程式 (8.15) p21 の一般化力 $\boldsymbol{\mathcal{F}}$ に，

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{F}}_T \tag{9.5}$$

として代入すれば，トルク T を受けた車輪型倒立ロボットの運動方程式が得られる．

この系を倒立状態で安定化するために，関節トルク T に，次のような **PD 制御入力** [7] を与えることにする．

$$T = K_1\alpha + K_2\dot{\alpha} + K_3\beta + K_4\dot{\beta} \tag{9.6}$$

これにより，原点 $x = r\alpha = 0$ に収束する位置制御と，倒立姿勢 $\beta = 0$ に収束する倒立制御が達成される．

実習 3.2 (車輪型倒立ロボットの倒立制御)　$(K_1, K_2, K_3, K_4) = (1, 0.5, 40, 4)$ のときのロボットの動きを確認せよ．このときロボットは原点で倒立安定化する．

▶ **解答例**　**Code 4** の 7 行目：

```
FT = [0; 0];
```

を，

```
T = 1*a + 0.5*da + 40*b + 4*db; FT = [T; -T];
```

に変更して実行すればよい．

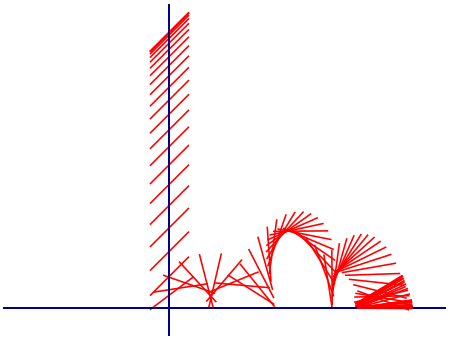
---

[6] 肩の T は転置の T.
[7] 機械力学 [1] 12.3 節を復習せよ．

10　接触と摩擦１（バウンド）

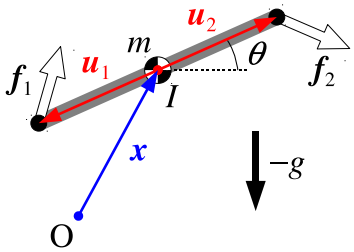
実例を見てしまうのがはやい．

実習 3.3　**Code 5** は，質量 m で長さが l の均一な剛体棒が，床面でバウンドするシミュレーションである．**Code 5** を実行せよ．



10.1　床面でバウンドする棒

Code 5 のシミュレータは，次の力学モデルを解いている．



$\boldsymbol{f}_1, \boldsymbol{f}_2$ は端点が受ける外力で，後に，床からの反力と摩擦力が代入される．$\boldsymbol{g}$ は重力加速度である．$\boldsymbol{x}$ は重心の位置ベクトル，θ は棒の傾斜角であり，

$$\boldsymbol{u}_1 = -\frac{l}{2}\begin{bmatrix}\cos\theta \\ \sin\theta\end{bmatrix}, \quad \boldsymbol{u}_2 = -\boldsymbol{u}_1 \tag{10.1}$$

は，重心 (この棒では中点) から見た $\boldsymbol{f}_1, \boldsymbol{f}_2$ の着力点の位置ベクトルである．棒が十分に細いと仮定すると，重心まわりの慣性モーメントは，$I = ml^2/3$ となる．

単純な系なので，解析力学ではなく，剛体の運動法則に基づいて運動方程式を求めよう．まず，重心で力とトルクの集約すると，この棒に作用する合力は，

$$\boldsymbol{F} = \boldsymbol{f}_1 + \boldsymbol{f}_2 + m\boldsymbol{g} \tag{10.2}$$

合トルクは，(“∧” は**符号付き面積**[8])

$$T = \boldsymbol{u}_1 \wedge \boldsymbol{f}_1 + \boldsymbol{u}_2 \wedge \boldsymbol{f}_2 \tag{10.3}$$

[8] 機械力学 [1] 3 章を復習せよ．

となる．これらを，剛体の運動方程式[9]に代入すると，

$$\begin{cases} m\ddot{\boldsymbol{x}} = \boldsymbol{F} = \boldsymbol{f}_1 + \boldsymbol{f}_2 + m\boldsymbol{g} & (\text{ニュートン方程式}) \\ J\ddot{\theta} = T = \boldsymbol{u}_1 \wedge \boldsymbol{f}_1 + \boldsymbol{u}_2 \wedge \boldsymbol{f}_2 & (\text{オイラー方程式}) \end{cases} \tag{10.4}$$

となる．これを解いてアニメーション表示したのが **Code 5** である．

ただし現時点では，外力 $\boldsymbol{f}_1, \boldsymbol{f}_2$ の具体形が不明である．以下，床からの反力と摩擦力を数式表現し，$\boldsymbol{f}_1, \boldsymbol{f}_2$ に代入する方法について述べる．

以下，外力 $\boldsymbol{f}_i$ を，垂直反力 $\boldsymbol{R}_i$ と，摩擦力 $\boldsymbol{F}_i$ の和として，

$$\boldsymbol{f}_i = \boldsymbol{F}_i + \boldsymbol{R}_i = \begin{bmatrix} F_i \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ R_i \end{bmatrix} \quad (i = 1, 2) \tag{10.5}$$

と表す．垂直抗力 $\boldsymbol{R}_i$ の具体形は **10.2 節**で，摩擦力 $\boldsymbol{F}_i$ の具体形は **11 節**で与える．

10.2　垂直抗力の数理モデル — ペナルティー法

自由落下する棒が，床に着く前までは，端点に受ける外力はないので $\boldsymbol{f}_1 = \boldsymbol{f}_2 = \mathbb{O}$ である．ところが，床と接触したとたん，棒は床から反力 $\boldsymbol{f}_1, \boldsymbol{f}_2 \neq \mathbb{O}$ を受ける．反力に弾かれた棒が，再び宙に舞うと，また $\boldsymbol{f}_1 = \boldsymbol{f}_2 = \mathbb{O}$ に戻る．こうした状況は，どのように数式表現できるだろうか？

簡単のため，図 3 の左に示すように，自由落下する 1 質点が床で反発する問題を考える．こうした反発現象のモデルとして，広く実用されているのが，図 3 の右である．床をばねとダンパーで表し，質点と床が接触する位置を $y = 0$ としている．こうしたトランポリン型のモデル化手法を，**ペナルティー法**という．

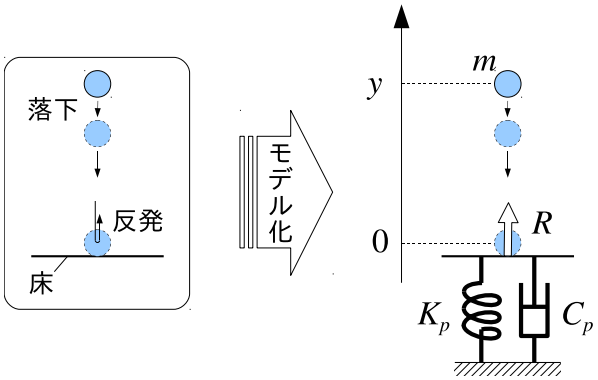


図 3: 垂直抗力のモデル (ペナルティー法)

このモデルでは，質点は床下までめり込む．めり込んでいる間だけ，ばねとダンパーによる反力 R が発生すると考える．式で書くと，

$$R = R(y, \dot{y}) = \begin{cases} 0 & (y > 0) \\ -K_p y - C_p \dot{y} & (y \leq 0) \end{cases} \tag{10.6}$$

[9] 機械力学 [1] 7 章の**ニュートン・オイラー方程式**を復習せよ．

である．K_p は質点が床にめり込んだときのばね定数，C_p は同じく粘性係数である．K_p で床の硬さを調整し，C_p で反発係数を調整する．また，プログラミングでは，(10.6) の場合分けをステップ関数で，次のように書くテクニックも使われる [10)]．

$$R = R(y, \dot{y}) = U(-y)\{-K_p y - C_p \dot{y}\} \tag{10.7}$$

関数 $U(X)$ を，**単位ステップ関数** (unit step function) と呼ぶ．定義は，

$$U(X) = \begin{cases} 1 & (X \geq 0) \\ 0 & (X < 0) \end{cases} \tag{10.8}$$

である．式 (10.7) のなかでは，単位ステップ関数を左右反転して使うために，$U(-y)$ としている．

以上のモデルでは，C_p の粘性抵抗が，田んぼに足を突っ込んだときのように，足を引き抜くときにも効く．これをトランポリンのように，引き抜く方向ではゼロと仮定して，

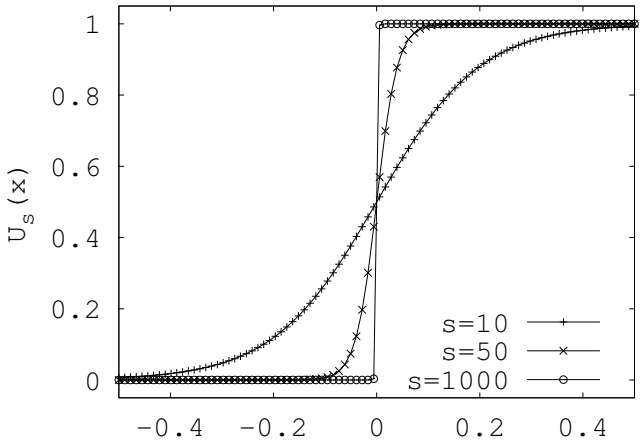
$$R = R(y, \dot{y}) = U(-y)\{-K_p y - U(-\dot{y})C_p \dot{y}\} \tag{10.9}$$

とすることもできる [11)]．

10.3 シグモイド関数の活用

その他，プログラミング上のテクニックとして，ステップ関数の代用に，ステップ関数の角を丸めたシグモイド関数：

$$U_s(X) := \frac{1}{1 + \exp(-sX)} \tag{10.10}$$



を使う場合がある．s はステップの丸みを表わすパラメータであり，s を大きくすると丸みが鋭くなる ($s \to \infty$ の極限でステップ関数となる)．この代用の理由だが，数値積分のアルゴリズムはステップ関数のような不連続関数が苦手であり，それが原因で数値積分に失敗することがある (滅茶苦茶な解がでる or エラーがでて解がでない)．このようなとき，シグモイド関数を使って，エラーが解消する程度まで，徐々にステップの角を丸めていく．もちろん，丸めすぎると，硬い床のモデルとはいえなくなるので，要注意．

[10)] **Code 5** p32 の「方法 1」
[11)] **Code 5** p32 の「方法 2」

10.4　床面で反発する棒の例題

式 (10.7)，(10.9)，(10.10) より，式 (10.5) における垂直抗力 $\boldsymbol{R}_i$ の成分 R_i は，

$$R_i = R(y_i, \dot{y}_i) = U_s(-y_i)\{-K_p y_i - C_p \dot{y}_i\} \quad (i = 1, 2) \tag{10.11}$$

または，

$$R_i = R(y_i, \dot{y}_i) = U_s(-y_i)\{-K_p y_i - U(-\dot{y}_i) C_p \dot{y}_i\} \quad (i = 1, 2) \tag{10.12}$$

となる．ただし，y_i は，端点の位置ベクトル $\boldsymbol{x}_i = \boldsymbol{x} + \boldsymbol{u}_i$ の y 成分，$\dot{y}_i$ は同じく速度ベクトル，

$$\dot{\boldsymbol{x}}_i = \dot{\boldsymbol{x}} + \dot{\boldsymbol{u}}_i \quad (i = 1, 2), \quad \dot{\boldsymbol{u}}_1 = \frac{l\dot{\theta}}{2} \begin{bmatrix} \sin\theta \\ -\cos\theta \end{bmatrix}, \ \dot{\boldsymbol{u}}_2 = -\dot{\boldsymbol{u}}_1 \tag{10.13}$$

の y 成分である．

11　接触と摩擦２（スリップ）

11.1　摩擦の数理モデル

物体を力 R で床に押し付けたときに発生する摩擦力は，

$$F = F(\dot{x}) = \begin{cases} -\mu R & (\dot{x} > 0) \\ 0 & (\dot{x} = 0) \\ \mu R & (\dot{x} < 0) \end{cases} \tag{11.1}$$

と書ける．$\dot{x}$ は床から見た物体の相対速度である．また，μ を物体と床の**動摩擦係数** (coefficient of dynamic friction) という．符号関数

$$\mathrm{sgn}(X) := \begin{cases} 1 & (X > 0) \\ 0 & (X = 0) \\ -1 & (X < 0) \end{cases} \tag{11.2}$$

を使うと，(11.1) と同じことは，

$$F = F(\dot{x}) = -\mu R \, \mathrm{sgn}\dot{x} \tag{11.3}$$

とも書ける．

11.2　シグモイド関数の活用

(11.1) または (11.3) は不連続関数なので，垂直抗力のときと同様に，数値計算上の問題を引き起す．同様の対処法として，シグモイド関数 (10.10) を使って，(11.3) の不連続部分を丸めるには，

$$F = F(\dot{x}) = -\mu R \, \mathrm{sgn}_s(\dot{x}), \quad \mathrm{sgn}_s(\dot{x}) := 2U_s(\dot{x}) - 1 \tag{11.4}$$

のようにすればよい．0 から 1 まで変化する U_s の高さを 2 倍して縦に -1 平行移動することで，-1 から 0 を経て 1 に至る関数 $\mathrm{sgn}_s(\dot{x})$ を作り出している．

11.3 床面で反発する棒の例題

式 (10.10)，(11.3)，(11.4) より，式 (10.5) における摩擦力 $\boldsymbol{F}_i$ の成分 F_i は，

$$F_i = F(\dot{x}_i, y_i, \dot{y}_i) = -\mu R(y_i, \dot{y}_i)\,\mathrm{sgn}_s(\dot{x}_i) \tag{11.5}$$

となる．ただし，$R(y_i, \dot{y}_i)$ は，(10.11)，(10.12) の垂直抗力である．また，$\dot{x}_i$ は端点の速度ベクトル $\dot{\boldsymbol{x}}_i = \dot{\boldsymbol{x}} + \dot{\boldsymbol{u}}_i$ の x 成分である．

以上，式 (10.11)，(10.12)，(11.5) をまとめると，床との接触によって棒の端点が受ける外力 $\boldsymbol{f}_i$ の数式表現が，次のように得られる．

$$\boldsymbol{f}_i = \begin{bmatrix} F_i \\ R_i \end{bmatrix} = \begin{bmatrix} -\mu R_i\,\mathrm{sgn}_s(\dot{x}_i) \\ U_s(-y_i)\{-K_p y_i - C_p \dot{y}_i\} \end{bmatrix} \qquad (i = 1, 2) \tag{11.6}$$

$$\text{または } = \begin{bmatrix} F_i \\ R_i \end{bmatrix} = \begin{bmatrix} -\mu R_i\,\mathrm{sgn}_s(\dot{x}_i) \\ U_s(-y_i)\{-K_p y_i - U(-\dot{y}_i) C_p \dot{y}_i\} \end{bmatrix} \qquad (i = 1, 2) \tag{11.7}$$

11.4 応用 — 転倒する車輪型倒立ロボットの作成

11.4.1 転倒に必要なカラクリ

フィードバック制御で安定化された倒立ロボットが転倒するには，ある限界で制御を諦めるカラクリが必要である．その一例として，振り子の傾斜角の絶対値 $|\beta|$ が基準値 $\beta_{\max} > 0$ を超えたときに制御が打ち切られるような，フィードバック制御入力，

$$T \equiv \begin{cases} K_1\alpha + K_2\dot{\alpha} + K_3\beta + K_4\dot{\beta} & (|\beta| \le \beta_{\max}) \\ 0 & (\text{それ以外}) \end{cases} \tag{11.8}$$

を考えることができる．

垂直抗力 (10.7) p28 のときと同様に，場合分けを関数で書くには，次のような**台形関数** (trapezoidal function) を使えばよい．台形関数とは，両側で 0，中央付近で 1 に立ち上がる台形状の関数のことである．これは，左右対称な 2 個のステップ関数 (10.8) p28 を，左右に平行移動してから掛け算することで，次のように構成できる．

$$\mathrm{trap}(x; x_0, w) \equiv U\big(-(x - x0) + w\big) \cdot U\big((x - x0) + w\big) \tag{11.9}$$

$x0$ は台形の中心，w は台形の幅の 1/2 である．これを使うと，(11.8) と同じことは，

$$T \equiv \mathrm{trap}(\beta; 0, \beta_{\max}) \cdot (K_1\alpha + K_2\dot{\alpha} + K_3\beta + K_4\dot{\beta}) \tag{11.10}$$

と書ける．これで数値積分が不安定になる場合は，ステップ関数 $U(x)$ の代りに，これを丸めた $U_s(x)$ を使えばよい．

実習 3.4 **Code 6** を実行し，(11.10) を制御入力とする台車型倒立ロボットの運動を観察せよ．指定された角度 b0 で制御が打ち切られ，自由運動に移行してスイングを続ける様子が見てとれる．

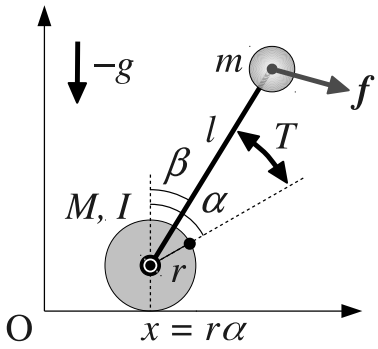
11.4.2　床面の設定

自由運動でスイングする振り子の先端を，**ペナルティー法** p27 で設けた床面にぶち当てれば，床面に倒れこむ動作がシミュレートできる．床面との衝突の際，振り子の先端は力，

$$\boldsymbol{f} = \begin{bmatrix} F \\ R \end{bmatrix} \tag{11.11}$$

を受ける．R は床からの垂直抗力，F は摩擦力である．

これを運動方程式に組み込むための枠組みとして，振り子の先端に力 $\boldsymbol{f} = (f_x, f_y)^T$ を作用させた次のモデルを導入する．簡単のため，棒や先端の太さは無視できるとしよう．



以上の $\boldsymbol{f}$ が角度 α, β に及ぼす一般化力 $\boldsymbol{\mathcal{F}}_{\boldsymbol{f}} \equiv (f_\alpha, f_\beta)^T$ を求めて，運動方程式 (8.15) p21 の $\mathcal{F}_\alpha, \mathcal{F}_\beta$ に加算すれば，このモデルの運動方程式が得られることになる．

まず，一般化座標 $\boldsymbol{q} = (\alpha, \beta)^T$ から，振り子先端の直交座標 $\boldsymbol{x}_m$ への座標変換は，

$$\boldsymbol{x}_m = \boldsymbol{x}_m(\alpha, \beta) = \begin{bmatrix} x_m(\alpha, \beta) \\ y_m(\alpha, \beta) \end{bmatrix} = \begin{bmatrix} r\alpha + l\sin\beta \\ r + l\cos\beta \end{bmatrix} \tag{8.4 p20}$$

であった．ゆえに，算法 3.1〜3.2 から変換公式：

$$\boldsymbol{\mathcal{F}}_{\boldsymbol{f}} = \begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix} = \begin{bmatrix} \frac{\partial x_m}{\partial \alpha} & \frac{\partial y_m}{\partial \alpha} \\ \frac{\partial x_m}{\partial \beta} & \frac{\partial y_m}{\partial \beta} \end{bmatrix} \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} r & 0 \\ l\cos\beta & -l\sin\beta \end{bmatrix} \boldsymbol{f} \tag{11.12}$$

を得る．最後に，右辺の $\boldsymbol{f}$ をペナルティー法 (11.6) p30 の要領で，

$$\boldsymbol{f} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} F \\ R \end{bmatrix} = \begin{bmatrix} -\mu R \,\mathrm{sgn}_s(\dot{x}_m) \\ U_s(-y)\{-K_p y_m - C_p \dot{y}_m\} \end{bmatrix} \tag{11.13}$$

のように与えれば，振り子の先端と床との衝突がシミュレートされる．ただし

$$\dot{\boldsymbol{x}}_m = \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \end{bmatrix} = \begin{bmatrix} r\dot{\alpha} + l\dot{\beta}\cos\beta \\ -l\dot{\beta}\sin\beta \end{bmatrix} \tag{8.5 p20}$$

である．

得られた床反力の効果 $\boldsymbol{\mathcal{F}}_{\boldsymbol{f}}$ を加算して，運動方程式 (8.15) p21 の一般化力 $\boldsymbol{\mathcal{F}}$ を

$$\boldsymbol{\mathcal{F}} = \underbrace{\boldsymbol{\mathcal{F}}_T}_{\text{倒立制御}} + \underbrace{\boldsymbol{\mathcal{F}}_{\boldsymbol{f}}}_{\text{床反力}} \tag{11.14}$$

とすれば，倒立ロボットの先端が床面に衝突する現象をシミュレートできる．

実習 3.5　**Code 7** を実行し，床の反力 (垂直抗力と摩擦力) を受ける台車型倒立ロボットの運動を観察せよ．指定された角度 `bmax` で制御が打ち切られて振り子の先端が床に落ち，バウンドして停止する [12]．

[12] K_p, C_p の調整が不十分で床にめり込みすぎかも．

参考文献

[1] 吉田勝俊著：「機械力学」（宇都宮大学大学生協）

A　プログラム例

Code 4: “wip.sce” (Scilab)

```
clear;clf();
///// 運動方程式の定義 /////
M=1, J=0.1, m=5, r=0.2, l=1, g=9.8;
function dx = eom(t,x)
    a=x(1); da=x(2); b=x(3); db=x(4);
//  T = 1*a + 0.5*da + 40*b + 4*db; FT = [T; -T];
    FT = [0; 0];
    A = [(M+m)*r^2+J, m*l*r*cos(b); ...
              m*l*r*cos(b),    m*l^2];
    bb = [m*l*r*db^2 *sin(b); m*g*l*sin(b)] + FT;
    h = A\bb;
    dx(1) = x(2); dx(2) = h(1);
    dx(3) = x(4); dx(4) = h(2);
endfunction
///// 運動方程式を解く /////
tn=1001; tt=linspace(0,0.01*tn,tn);
x0=[0; 0; 0.2; 0];
xx=ode( x0, 0, tt, eom );
///// ロボット描画用の関数 /////
function R=Rot(a) //回転
 R=[cos(a),-sin(a);...
    sin(a), cos(a)];
endfunction
function yy=Trans(xx,rr) //平行移動
 yy(1,:)=xx(1,:) + rr(1)*ones(xx(1,:));
 yy(2,:)=xx(2,:) + rr(2)*ones(xx(2,:));
endfunction
function draw_robot(a,b) //ロボット描画
    x=r*a; //車輪中心の水平変位
    sn=5; sq=linspace(0,2*%pi*(sn-1)/sn,sn);
    spoke=r*[cos(sq);sin(sq)];       //原点にあるスポークの外周点
    spoke=Rot(-a) * spoke;           //角度a回転したスポークの外周点
    spoke=Trans(spoke,[x;r]);        //位置xのスポークの外周点
    xv=[x*ones(1:sn);spoke(1,:)]; yv=[r*ones(1:sn);spoke(2,:)];
    xsegs(xv,yv,2); xarc(x-r, 2*r, 2*r, 2*r, 0, 360*64); //スポーク; 車輪
    rod=[0,0;0,l];                   //原点にある棒の両端点
    rod=Rot(-b) * rod;               //角度βだけ回転した棒の両端点
    rod=Trans(rod,[x;r]);            //位置xの棒の両端点
    plot( rod(1,:), rod(2,:),"r-"); //棒の描画
    p=gce(); p.children.thickness=2;//直前の描画; 線の太さ
    g=gca(); g.data_bounds=[-4,-1;4,2]; //座標軸の範囲
    g.isoview="on"; xgrid(4);            //縦横比1; グリッド;
endfunction
a=x0(1); b=x0(3); drawlater; draw_robot(a,b); drawnow;
sleep(2000); //2s待ち
realtimeinit(0.01); //アニメーションの時間刻み
for i=1:10:tn
    realtime(i);
    drawlater; clf();        //描画延期; 描画消去;
    a=xx(1,i); b=xx(3,i);
    draw_robot(a,b);
    xlabel(sprintf("%d / %d", i, tn));
    drawnow;                 //画面更新;
end
```

Code 5: “stick.sce” (Scilab)

```
clear;clf();
///// 運動方程式 /////
m=1; l=1; J=m*l^2/3;  g = 9.8; mu=0.3;
function y = torq(x,f)
    y = det([x,f]);
endfunction
function y = signum( x )
    y = (2.0/(1+exp(-1e4*x))-1.0); //符号関数
endfunction
```

```
function y = step( x )
    y = 1.0/(1.0+exp(-1e4*x)); //ステップ関数
endfunction
function fy = R( y, dy ) //床反力
    Kp=8e3; Cp=25; fy = step(-y)*(-Kp*y -Cp*dy);              //『方法1』
    //Kp=12e3; Cp=80; fy = step(-y)*(-Kp*y -step(-dy)*Cp*dy); //『方法2』
endfunction
function fx = F( dx, R ) //摩擦力
    fx = -signum(dx)*mu*R;
endfunction
function dq = eom(t,q)
    x=q(1); dx=q(2); y=q(3); dy=q(4); a=q(5); da=q(6);
    u1 = -(l/2)*[cos(a); sin(a)]; u2 = -u1;
    du1 =-(l/2)*da*[-sin(a); cos(a)]; du2 = -du1;
    X1 = [x;y]+u1;          //端点1の位置
    X2 = [x;y]+u2;          //端点2の位置
    dX1 = [dx;dy]+du1;      //端点1の速度
    dX2 = [dx;dy]+du2;      //端点2の速度
    R1 = R(X1(2), dX1(2)); //床垂直抗力1
    R2 = R(X2(2), dX2(2)); //床垂直抗力2
    F1 = F(dX1(1), R1);     //床摩擦力1
    F2 = F(dX2(1), R2);     //床摩擦力2
    f1 = [F1;R1]; f2 = [F2;R2];         //端点に作用する力
    F = f1 + f2 + m*[0;-g];             //合力
    T = torq(u1, f1) + torq(u2, f2);    //合トルク
    dq(1) = dx; dq(2) = F(1)/m;
    dq(3) = dy; dq(4) = F(2)/m;
    dq(5) = da; dq(6) = T/J;
endfunction
///// 運動方程式を解く /////
n=130; tt=linspace(0,0.05*(n-1),n);
x0 = [0; 0; 5; 0; %pi/4; 0];
xx=ode( x0, 0, tt, eom );
///// アニメーションする /////
function draw_mech(x,y,th)
    g=gca();                        //座標軸の取得
    g.data_bounds(:,1)=[-2;4]; //x軸の範囲
    g.data_bounds(:,2)=[0;5];  //y軸の範囲
    g.isoview="on";             //縦横比1; グリッド;
    xset("color",1);            //基本色黒
    xsegs([-3;5],[0;0],9);      //x軸
    xsegs([0;0],[-0.5;5.5],9); //y軸
    u1 = -(l/2)*[cos(th); sin(th)]; u2 = -u1;
    X1 = [x;y]+u1; X2 = [x;y]+u2;
    xsegs([X1(1);X2(1)],[X1(2);X2(2)],5); //スティック
    p=gce(); p.thickness=2;     //直前の描画; 線の太さ
endfunction
x=x0(1); y=x0(3); th=x0(5);
draw_mech(x,y,th); drawnow; //描画実行; 画面更新;
//xclick(); //マウスクリック待ち
sleep(2000);
realtimeinit(0.05); //アニメーションの時間刻み
for i=1:n
    realtime(i);
    drawlater(); clf(); //描画延期; 描画消去;
    x=xx(1,i); y=xx(3,i); th=xx(5,i);
    draw_mech(x,y,th);  //描画消去; 棒を描く;
    drawnow;            //画面更新
end
//xs2eps(0,"stick.eps");
```

Code 6: “wip-fall-free.sce” (Scilab)

```
clear;clf();
/////////////// 諸元 ///////////////
M=1, J=0.1, m=5, r=0.2, l=1, g=9.8;
bmax=0.29; //転倒角
////////////////////////////////////
///// ステップ関数ほか /////
function y = step( x )
    s = 100;
    y = 1.0./(1.0+exp(-s*x));               //ステップ関数
endfunction
function y = trap( x, x0, w )
    y = step((x-x0)+w).*step(-(x-x0)+w);    //台形関数
endfunction
///// 運動方程式の定義 /////
function dx = eom(t,x)
    a=x(1); da=x(2); b=x(3); db=x(4);
    T = trap(b,0,bmax)*(1*a + 0.5*da + 40*b + 4*db); //制御の打ち切り
```

```
    FT = [T; -T];   //トルク→一般化力
    A = [(M+m)*r^2+J, m*l*r*cos(b); ...
              m*l*r*cos(b),    m*l^2];
    bb = [m*l*r*db^2 *sin(b); m*g*l*sin(b)] + FT;
    h = A\bb;       //加速度の右辺
    dx(1) = x(2); dx(2) = h(1);
    dx(3) = x(4); dx(4) = h(2);
endfunction
///// 運動方程式を解く /////
tn=1001; tt=linspace(0,0.01*tn,tn);
x0=[0; 0; 0; 1];
xx=ode( x0, 0, tt, eom );
///// ロボット描画用の関数 /////
function R=Rot(a)           //回転
    R=[cos(a),-sin(a);...
       sin(a), cos(a)];
endfunction
function yy=Trans(xx,rr)    //平行移動
    yy(1,:)=xx(1,:) + rr(1)*ones(xx(1,:));
    yy(2,:)=xx(2,:) + rr(2)*ones(xx(2,:));
endfunction
function draw_robot(a,b)    //ロボット描画
    x=r*a; //車輪中心の水平変位
    sn=5; sq=linspace(0,2*%pi*(sn-1)/sn,sn);
    spoke=r*[cos(sq);sin(sq)];      //原点にあるスポークの外周点
    spoke=Rot(-a) * spoke;          //角度a回転したスポークの外周点
    spoke=Trans(spoke,[x;r]);       //位置xのスポークの外周点
    xv=[x*ones(1:sn);spoke(1,:)]; yv=[r*ones(1:sn);spoke(2,:)];
    xsegs(xv,yv,2); xarc(x-r, 2*r, 2*r, 2*r, 0, 360*64); //スポーク; 車輪
    rod=[0,0;0,l];                  //原点にある棒の両端点
    rod=Rot(-b) * rod;              //角度βだけ回転した棒の両端点
    rod=Trans(rod,[x;r]);           //位置xの棒の両端点
    plot( rod(1,:), rod(2,:),"r-"); //棒の描画
    p=gce(); p.children.thickness=2;    //直前の描画; 線の太さ
    g=gca(); g.data_bounds=[-4,-1;4,2]; //座標軸の範囲
    g.isoview="on"; xgrid(4);           //縦横比1; グリッド;
endfunction
a=x0(1); b=x0(3); drawlater; draw_robot(a,b); drawnow;
sleep(2000); //2s待ち
realtimeinit(0.01); //アニメーションの時間刻み
for i=1:10:tn
    realtime(i);
    drawlater; clf();       //描画延期; 描画消去;
    a=xx(1,i); b=xx(3,i);
    draw_robot(a,b);
    xlabel(sprintf("%d / %d", i, tn));
    drawnow;                //画面更新;
end
```

Code 7: “wip-fall-floor.sce” (Scilab)

```
clear;clf();
/////////////// 諸元 ///////////////
M=1, J=0.1, m=5, r=0.2, l=1, g=9.8;
bmax=0.29; //転倒角
mu=0.3; //摩擦係数
////////////////////////////////////
///// ステップ関数ほか /////
function y = step( x )
 s = 100;
 y = 1.0./(1.0+exp(-s*x));              //ステップ関数
endfunction
function y = trap( x, x0, w )
 y = step((x-x0)+w).*step(-(x-x0)+w);   //台形関数
endfunction
function y = sgn( x )
    s = 1e4;
    y = (2.0/(1+exp(-s*x))-1.0);        //符号関数
endfunction
///// 床からの反力（ペナルティー法） /////
function ff = yuka(x)
  global r l mu;
  a=x(1); da=x(2); b=x(3); db=x(4);
  xm = [r*a+l*sin(b); r+l*cos(b)];
  dxm = [r*da+l*db*cos(b); -l*db*sin(b)];
  Kp=1e4; Cp=90;    //これらの数値を変えると床の性質が変わる
  R = step(-xm(2))*(-Kp*xm(2) -Cp*dxm(2)); //垂直抗力
  F = -mu*R*sgn(dxm(1)); //摩擦力
  ff = [F; R];      //床からの反力
endfunction
///// 運動方程式の定義 /////
```

```
function dx = eom(t,x)
    a=x(1); da=x(2); b=x(3); db=x(4);
    /// 制御の打ち切り×(位置制御 + 倒立制御) ///
    T = trap(b,0,bmax)*(1*a + 0.5*da + 40*b + 4*db);
    FT = [T; -T];    //トルク→一般化力
    /// ペナルティー法による床反力 ///
    ff=yuka([a; da; b; db]);
    Ff = [r, 0; l*cos(b), -l*sin(b)]*ff; //床反力→一般化力
    A = [(M+m)*r^2+J, m*l*r*cos(b); ...
            m*l*r*cos(b),    m*l^2];
    bb = [m*l*r*db^2 *sin(b); m*g*l*sin(b)] + FT + Ff;
    h = A\bb;        //加速度の右辺
    dx(1) = x(2); dx(2) = h(1);
    dx(3) = x(4); dx(4) = h(2);
endfunction
///// 運動方程式を解く /////
tn=401; tt=linspace(0,0.01*tn,tn);
x0=[0; 0; 0; 1];
xx=ode( x0, 0, tt, eom );
///// ロボット描画用の関数 /////
function R=Rot(a)            //回転
 R=[cos(a),-sin(a);...
    sin(a), cos(a)];
endfunction
function yy=Trans(xx,rr)     //平行移動
 yy(1,:)=xx(1,:) + rr(1)*ones(xx(1,:));
 yy(2,:)=xx(2,:) + rr(2)*ones(xx(2,:));
endfunction
function draw_robot(a,b)     //ロボット描画
    x=r*a; //車輪中心の水平変位
    sn=5; sq=linspace(0,2*%pi*(sn-1)/sn,sn);
    spoke=r*[cos(sq);sin(sq)];  //原点にあるスポークの外周点
    spoke=Rot(-a) * spoke;      //角度a回転したスポークの外周点
    spoke=Trans(spoke,[x;r]);   //位置xのスポークの外周点
    xv=[x*ones(1:sn);spoke(1,:)]; yv=[r*ones(1:sn);spoke(2,:)];
    xsegs(xv,yv,2); xarc(x-r, 2*r, 2*r, 2*r, 0, 360*64); //スポーク; 車輪
    rod=[0,0;0,l];              //原点にある棒の両端点
    rod=Rot(-b) * rod;          //角度βだけ回転した棒の両端点
    rod=Trans(rod,[x;r]);       //位置xの棒の両端点
    plot( rod(1,:), rod(2,:),"r-");     //棒の描画
    p=gce(); p.children.thickness=2;    //直前の描画; 線の太さ
    g=gca(); g.data_bounds=[-4,-1;4,2]; //座標軸の範囲
    g.isoview="on"; xgrid(4);           //縦横比1; グリッド;
endfunction
a=x0(1); b=x0(3); drawlater; draw_robot(a,b); drawnow;
sleep(2000); //2s待ち
realtimeinit(0.01); //アニメーションの時間刻み
for i=1:10:tn
    realtime(i);
    drawlater; clf();   //描画延期; 画面消去;
    a=xx(1,i); b=xx(3,i);
    draw_robot(a,b);
    xlabel(sprintf("%d / %d", i, tn));
    drawnow;            //画面更新;
end
```