13

ロボットシミュレータ

9.5 節のような力学シミュレータに，床面などの外部環境を追加したい．そのために必要なのは，衝突や摩擦のカラクリである．これらをシミュレータに追加すると，ロボットの跳躍や転倒，ロボット間の衝突などの状況が再現できる．

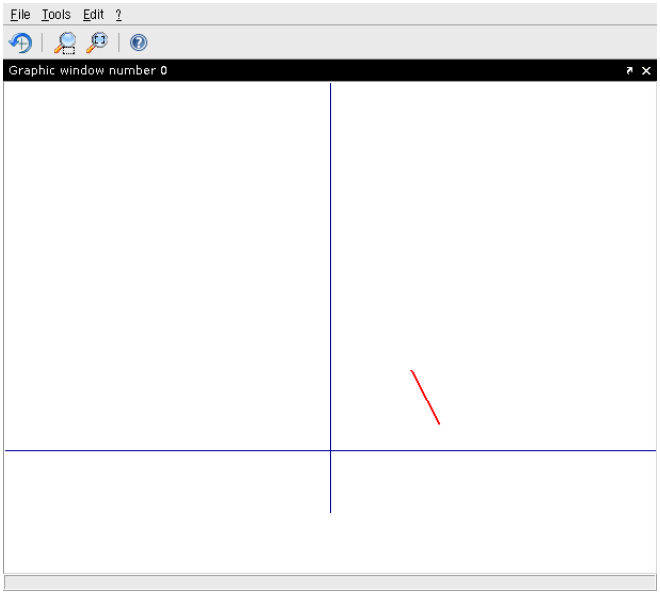
13.1 床面で反発する棒の運動

実例を見てしまうのがはやい．以下は，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．実行せよ．

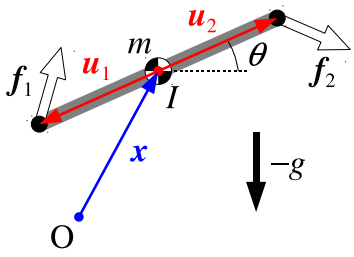
Code 21 “stick.sci” (Scilab)

```
clear;clf(); set figure_style new;
//運動方程式を解く
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
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)=[-3;3]; //x軸の範囲
 g.data_bounds(:,2)=[-1;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([-10;10],[0;0],9);        //x軸
 xsegs([0;0],[-1;11],9);         //y軸
 u1 = -(1/2)*[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
realtimeinit(0.05); //アニメーションの時間刻み
for i=1:n
 realtime(i);
 drawlater();          //描画延期
 x=xx(1,i); y=xx(3,i); th=xx(5,i);
 clf(); draw_mech(x,y,th); //描画消去; 棒を描く;
 drawnow;              //画面更新
end
```



このシミュレータは，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，

$$\boldsymbol{u}_1 = -\frac{l}{2}\begin{bmatrix}\cos\theta \\ \sin\theta\end{bmatrix}, \quad \boldsymbol{u}_2 = -\boldsymbol{u}_1 \tag{13.1}$$

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，力学法則 7.1 p67 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，

$$\boldsymbol{F} = \boldsymbol{f}_1 + \boldsymbol{f}_2 + m\boldsymbol{g} \tag{13.2}$$

合トルクは，(“$\wedge$” は符号付き面積 p27)

$$T = \boldsymbol{u}_1 \wedge \boldsymbol{f}_1 + \boldsymbol{u}_2 \wedge \boldsymbol{f}_2 \tag{13.3}$$

となる．これらを力学法則 7.1 p67 に代入すると，運動方程式は，

$$\begin{cases} m\ddot{\boldsymbol{x}} = \boldsymbol{F} = \boldsymbol{f}_1 + \boldsymbol{f}_2 + m\boldsymbol{g} & \text{(ニュートン方程式)} \\ J\ddot{\theta} = T = \boldsymbol{u}_1 \wedge \boldsymbol{f}_1 + \boldsymbol{u}_2 \wedge \boldsymbol{f}_2 & \text{(オイラー方程式)} \end{cases} \tag{13.4}$$

となる．これを解いてアニメーション表示したのが **Code 21** 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，

$$\boldsymbol{f}_i = \boldsymbol{F}_i + \boldsymbol{R}_i = \begin{bmatrix}F_i \\ 0\end{bmatrix} + \begin{bmatrix}0 \\ R_i\end{bmatrix} \quad (i = 1, 2) \tag{13.5}$$

と表す．垂直抗力 $\boldsymbol{R}_i$ の具体形は **13.2** 節で，摩擦力 $\boldsymbol{F}_i$ の具体形は **13.3** 節で与える．

13.2 垂直抗力のモデル

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？

13.2.1　ペナルティー法

簡単のため，図 13.1 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 13.1 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．

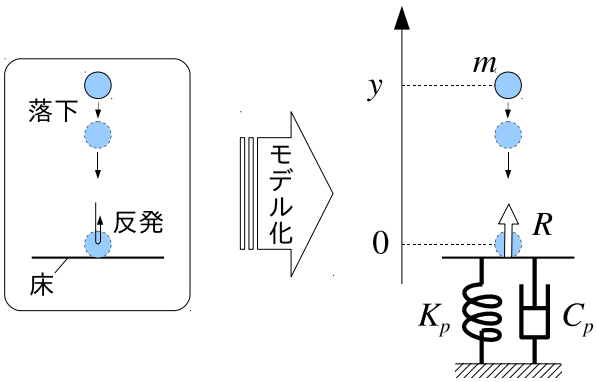


図 13.1　垂直抗力のモデル (ペナルティー法)

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，

$$R = R(y, \dot{y}) = \begin{cases} 0 & (y > 0) \\ -K_p y - C_p \dot{y} & (y \leq 0) \end{cases} \tag{13.6}$$

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，(13.6) 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[1)]．

$$R = R(y, \dot{y}) = U(-y)\{-K_p y - C_p \dot{y}\} \tag{13.7}$$

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

$$U(X) = \begin{cases} 1 & (X \geq 0) \\ 0 & (X < 0) \end{cases} \tag{13.8}$$

である．式 (13.7) 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．

[1)] **Code 21** p157 の「方法 1」

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，

$$R = R(y, \dot{y}) = U(-y)\{-K_p y - U(-\dot{y})C_p \dot{y}\} \tag{13.9}$$

とすることもできる[2)]．

13.2.2　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：

$$U_\alpha(X) := \frac{1}{1 + \exp(-\alpha X)} \tag{13.10}$$

を使う場合がある．α はステップの丸みを表わすパラメータであり，α を大きくすると丸みが鋭くなる ($\alpha \to \infty$ 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，それが原因のエラーを引き起す．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，反発のモデルとはいえなくなるので，要注意．

13.2.3　床面で反発する棒の例題

式 (13.7)，(13.9)，(13.10) より，式 (13.5) における垂直抗力 $\boldsymbol{R}_i$ の成分 R_i は，

$$R_i = R(y_i, \dot{y}_i) = U_\alpha(-y_i)\{-K_p y_i - C_p \dot{y}_i\} \quad (i = 1, 2) \tag{13.11}$$

または，

$$R_i = R(y_i, \dot{y}_i) = U_\alpha(-y_i)\{-K_p y_i - U(-y_i)C_p \dot{y}_i\} \quad (i = 1, 2) \tag{13.12}$$

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，

$$\dot{\boldsymbol{x}}_i = \dot{\boldsymbol{x}} + \dot{\boldsymbol{u}}_i \quad (i = 1, 2), \quad \dot{\boldsymbol{u}}_1 = \frac{l\dot{\theta}}{2}\begin{bmatrix} \sin\theta \\ -\cos\theta \end{bmatrix}, \ \dot{\boldsymbol{u}}_2 = -\dot{\boldsymbol{u}}_1 \tag{13.13}$$

の y 成分である．

[2)] **Code 21** p157 の「方法 2」

13.3　摩擦のモデル

13.3.1　摩擦の数式表現

物体を力 R で床に押し付けたときに発生する摩擦力は，

$$F = F(\dot{x}) = \begin{cases} -\mu R & (\dot{x} > 0) \\ 0 & (\dot{x} = 0) \\ \mu R & (\dot{x} < 0) \end{cases} \tag{13.14}$$

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

$$\mathrm{sgn}(X) := \begin{cases} 1 & (X > 0) \\ 0 & (X = 0) \\ -1 & (X < 0) \end{cases} \tag{13.15}$$

を使うと，(13.14) と同じことは，

$$F = F(\dot{x}) = -\mu R\,\mathrm{sgn}\dot{x} \tag{13.16}$$

とも書ける．

13.3.2　シグモイド関数の活用

(13.14) または (13.16) 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 (13.10) を使って，(13.16) の不連続部分を丸めるには，

$$F = F(\dot{x}) = -\mu R\,\mathrm{sgn}_\alpha(\dot{x}), \quad \mathrm{sgn}_\alpha(\dot{x}) := 2U_\alpha(\dot{x}) - 1 \tag{13.17}$$

のようにすればよい．0 から 1 まで変化する U_α 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 $\mathrm{sgn}_\alpha(\dot{x})$ を作り出している．

13.3.3　床面で反発する棒の例題

式 (13.10)，(13.16)，(13.17) より，式 (13.5) における摩擦力 $\boldsymbol{F}_i$ の成分 F_i は，

$$F_i = F(\dot{x}_i, y_i, \dot{y}_i) = -\mu R(y_i, \dot{y}_i)\,\mathrm{sgn}_\alpha(\dot{x}_i) \tag{13.18}$$

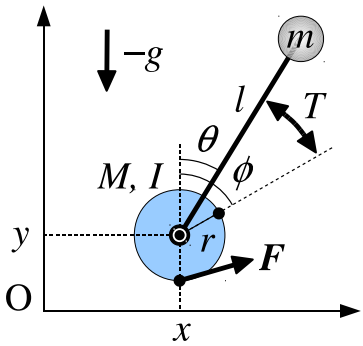
となる．ただし，$R(y_i, \dot{y}_i)$ は，(13.11)，(13.12) の垂直抗力である．また，$\dot{x}_i$ は端点の速度ベクトル $\dot{\boldsymbol{x}}_i = \dot{\boldsymbol{x}} + \dot{\boldsymbol{u}}_i$ の x 成分である．

以上，式 (13.11)，(13.12)，(13.18) 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．

$$\boldsymbol{f}_i = \begin{bmatrix} F_i \\ R_i \end{bmatrix} = \begin{bmatrix} -\mu R_i \operatorname{sgn}_\alpha(\dot{x}_i) \\ U_\alpha(-y_i)\{-K_p y_i - C_p \dot{y}_i\} \end{bmatrix} \qquad (i = 1, 2) \qquad (13.19)$$

$$\text{または} = \begin{bmatrix} F_i \\ R_i \end{bmatrix} = \begin{bmatrix} -\mu R_i \operatorname{sgn}_\alpha(\dot{x}_i) \\ U_\alpha(-y_i)\{-K_p y_i - U(-y_i) C_p \dot{y}_i\} \end{bmatrix} \qquad (i = 1, 2) \qquad (13.20)$$

13.4 演習 — 車輪型倒立ロボット



13.4.1 運動方程式

一般化座標を (x, y, ϕ, θ) とすると，座標変換は，

$$\boldsymbol{x}_M = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \boldsymbol{x}_m = \boldsymbol{x}_M + l \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} \qquad (13.21)$$

となる．運動エネルギーは，

$$\dot{\boldsymbol{x}}_M = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}, \quad \dot{\boldsymbol{x}}_m = \dot{\boldsymbol{x}}_M + l\dot{\theta} \begin{bmatrix} \cos\theta \\ -\sin\theta \end{bmatrix} \qquad (13.22)$$

より，

$$\begin{aligned} T &= \frac{M}{2}\dot{\boldsymbol{x}}_M^2 + \frac{m}{2}\dot{\boldsymbol{x}}_m^2 + \frac{I}{2}\dot{\phi}^2 \\ &= \frac{M}{2}(\dot{x}^2 + \dot{y}^2) + \frac{m}{2}\left((\dot{x} + l\dot{\theta}\cos\theta)^2 + (\dot{y} - l\dot{\theta}\sin\theta)^2\right) + \frac{I}{2}\dot{\phi}^2 \\ &= \frac{M}{2}(\dot{x}^2 + \dot{y}^2) + \frac{m}{2}\left(\dot{x}^2 + 2l\dot{\theta}\dot{x}\cos\theta + \dot{y}^2 - 2l\dot{\theta}\dot{y}\sin\theta + l^2\dot{\theta}^2\right) + \frac{I}{2}\dot{\phi}^2 \\ &= \frac{M+m}{2}(\dot{x}^2 + \dot{y}^2) + ml\dot{\theta}\,(\dot{x}\cos\theta - \dot{y}\sin\theta) + \frac{ml^2}{2}\dot{\theta}^2 + \frac{I}{2}\dot{\phi}^2 \end{aligned} \qquad (13.23)$$

となり，ポテンシャルは，

$$U = (M+m)gy + mgl\cos\theta + C \quad (C \text{ は定数}), \qquad (13.24)$$

となる．以上より，この系のラグランジュ関数 $L = T - U$ が定まる．

次に，一般化力を求めるために，$\boldsymbol{F} = (F_1, F_2)^T$ の着力点 $\boldsymbol{x}_F$ を求めると，

$$\boldsymbol{x}_F = \begin{bmatrix} x_{F1} \\ x_{F2} \end{bmatrix} = \begin{bmatrix} x + r\sin\phi \\ y + r\cos\phi \end{bmatrix}_{\phi=\pi} \tag{13.25}$$

となる．

▶▶ (車輪の一般化力)　式 (13.25) に $\phi = \pi$ を，あらかじめ代入して，整理してはいけない．あくまで，$\boldsymbol{F}$ は車輪の外周上に作用する力であり，たまたま地面の都合で，それが $\phi = \pi$ に位置すると考える．さもないと，$\boldsymbol{F}$ による ϕ の変化が記述できない．

この系が $\boldsymbol{F}$ から受ける一般化力 $\boldsymbol{\mathcal{F}}'$ は，算法 9.2 p106 より，

$$\begin{aligned}
\boldsymbol{\mathcal{F}}' = \begin{bmatrix} \mathcal{F}'_x \\ \mathcal{F}'_y \\ \mathcal{F}'_\phi \\ \mathcal{F}'_\theta \end{bmatrix} &= \begin{bmatrix} \frac{\partial x_{F1}}{\partial x} & \frac{\partial x_{F2}}{\partial x} \\ \frac{\partial x_{F1}}{\partial y} & \frac{\partial x_{F2}}{\partial y} \\ \frac{\partial x_{F1}}{\partial \phi} & \frac{\partial x_{F2}}{\partial \phi} \\ \frac{\partial x_{F1}}{\partial \theta} & \frac{\partial x_{F2}}{\partial \theta} \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ r\cos\phi & -r\sin\phi \\ 0 & 0 \end{bmatrix}_{\phi=\pi} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ -rF_1 \\ 0 \end{bmatrix}
\end{aligned} \tag{13.26}$$

となる．これに，関節トルク T を加算すると，この系が受ける一般化力は，

$$\boldsymbol{\mathcal{F}} = \begin{bmatrix} \mathcal{F}_x \\ \mathcal{F}_y \\ \mathcal{F}_\phi \\ \mathcal{F}_\theta \end{bmatrix} = \boldsymbol{\mathcal{F}}' + \begin{bmatrix} 0 \\ 0 \\ T \\ -T \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 + T \\ -rF_1 - T \\ 0 \end{bmatrix} \tag{13.27}$$

となる．

以上に求めた $L, \boldsymbol{\mathcal{F}}$ を，ラグランジュの運動方程式

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \mathcal{F}_q \quad (q = x, y, \phi, \theta) \tag{13.28}$$

に代入すると，次の運動方程式が得られる．

$$\begin{bmatrix} M+m & 0 & 0 & ml\cos\theta \\ 0 & M+m & 0 & -ml\sin\theta \\ 0 & 0 & I & 0 \\ ml\cos\theta & -ml\sin\theta & 0 & ml^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\phi} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} ml\dot{\theta}^2\sin\theta \\ -(M+m)g + ml\dot{\theta}^2\cos\theta \\ 0 \\ mgl\sin\theta \end{bmatrix} + \begin{bmatrix} \mathcal{F}_x \\ \mathcal{F}_y \\ \mathcal{F}_\phi \\ \mathcal{F}_\theta \end{bmatrix} \tag{13.29}$$

Maxima によるプログラム例と実行例を示す．

Code 22 “invwheel2-eom.mac” (Maxima)

```
xM:[x(t),y(t)];
/*xM:subst(r*p(t),x(t),xM);*/
xm:xM + l*[sin(q(t)),cos(q(t))];
DxM:diff(xM,t); Dxm:diff(xm,t);
Dx:diff(x(t),t); Dy:diff(y(t),t);
Dp:diff(p(t),t); Dq:diff(q(t),t);
Tene:(M/2)*(DxM.DxM) +(m/2)*(Dxm.Dxm) +(I/2)*(Dp^2), trigreduce, ratsimp;
Kene:M*g*xM[2] +m*g*xm[2], ratsimp;
L:Tene - Kene;
eqnx:diff(diff(L,Dx),t)-diff(L,x(t))-Fx,ratsimp;
eqny:diff(diff(L,Dy),t)-diff(L,y(t))-Fy,ratsimp;
eqnp:diff(diff(L,Dp),t)-diff(L,p(t))-Fp,ratsimp;
eqnq:diff(diff(L,Dq),t)-diff(L,q(t))-Fq,ratsimp;
eqn:[eqnx,eqny,eqnp,eqnq];
eqn:subst(dx,Dx,eqn); eqn:subst(dy,Dy,eqn);
eqn:subst(dp,Dp,eqn); eqn:subst(dq,Dq,eqn);
eqn:subst(ddx,diff(x(t),t,2),eqn);
eqn:subst(ddy,diff(y(t),t,2),eqn);
eqn:subst(ddp,diff(p(t),t,2),eqn);
eqn:subst(ddq,diff(q(t),t,2),eqn);
A:transpose(matrix(diff(eqn,ddx),diff(eqn,ddy),
  diff(eqn,ddp),diff(eqn,ddq)));
bb:A.[ddx,ddy,ddp,ddq]-eqn,ratsimp;
```

実行例 10 (Maxima)

```
> maxima -qb invwheel-eom2.mac
(((  筝        )))
(%i26) A : transpose(matrix(diff(eqn, ddx), diff(eqn, ddy),
                                    diff(eqn, ddp), diff(eqn, ddq)))
            [     M + m             0         0   l m cos(q(t))  ]
            [                                                    ]
            [       0             M + m       0  - l m sin(q(t)) ]
(%o26)      [                                                    ]
            [       0               0         I         0        ]
            [                                                    ]
            [                                           2        ]
            [ l m cos(q(t))  - l m sin(q(t))  0       l  m       ]
(%i27)         ev(bb : A . [ddx, ddy, ddp, ddq] - eqn, ratsimp)
                   [          2                          ]
                   [        dq  l m sin(q(t)) + Fx       ]
                   [                                     ]
                   [           2                         ]
(%o27)             [ - g M + dq  l m cos(q(t)) - g m + Fy ]
                   [                                     ]
                   [                  Fp                 ]
                   [                                     ]
                   [         g l m sin(q(t)) + Fq        ]
```

13.4.2 床との接触と摩擦

まず，車輪の下死点の位置ベクトルと速度ベクトルは，(13.25) より，

$$\boldsymbol{X} = \begin{bmatrix} x + r\sin\phi \\ y + r\cos\phi \end{bmatrix}_{\phi=\pi} = \begin{bmatrix} x \\ y - r \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{13.30}$$

である．ゆえに，車輪が床から受ける垂直抗力 R は，(13.7)，(13.10) より，

$$R = U_\alpha(-y)\{-K_p - C_p\dot{y}\} \qquad = F_2 = \mathcal{F}_y \tag{13.31}$$

と書ける．次に，摩擦力は，車輪外周と床との相対速度 $\Delta\dot{x}$ によって生じると仮定できる．車輪が回転しない ($\dot{\phi} = 0$) ときは $\Delta\dot{x} = \dot{x}$ である．車輪が回転すると ($\dot{\phi} > 0$)

相対速度は $r\dot{\phi}$ だけ減少するので，

$$\Delta\dot{x} = \dot{x} - r\dot{\phi} \tag{13.32}$$

となる．ゆえに，車輪が床から受ける摩擦力 F は，(13.17) より，

$$F = -\mu R\,\mathrm{sgn}_\alpha(\Delta\dot{x}) = -\mu\mathcal{F}_y\,\mathrm{sgn}_\alpha(\dot{x} - r\dot{\phi}) \qquad = F_1 = \mathcal{F}_x \tag{13.33}$$

と書ける．

13.4.3　倒立制御

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 制御入力を与えることにする．

$$T = K_1 x + K_2\dot{x} + K_3\theta + K_4\dot{\theta} \tag{13.34}$$

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

Scilab によるシミュレータを次に示す．摩擦係数を低めに設定したので，車輪がスリップする挙動が観察できる．

Code 23　“invwheel2.sci” (Scilab)

```
clear;clf(); set figure_style new;
//運動方程式を解く
M=1; r=0.2; m=5; l=1; J=0.1; g=9.8; mu=0.09;
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
endfunction
function fx = F( dx, R ) //摩擦力
 fx = -signum(dx)*mu*R;
endfunction
function dX = eom(t,X)
  x=X(1); dx=X(2); y=X(3); dy=X(4);
  p=X(5); dp=X(6); q=X(7); dq=X(8);
  Fy = R(y-r, dy); Fx = F(dx-r*dp, Fy); //垂直抗力; 摩擦力;
  Fp=-r*Fx; Fq=0;                       //φ θ 方向の一般化力
  T = 2.5*x+2*dx+ 50*q + 10*dq;         //制御トルクT
  Fp = Fp +T; Fq = Fq -T;
  A = [M+m, 0, 0, m*l*cos(q); ...
       0, M+m, 0,-m*l*sin(q); ...
       0, 0, J, 0; ...
       m*l*cos(q), -m*l*sin(q), 0, m*l^2];
  bb = [m*l*dq^2*sin(q)+Fx; ...
        -(M+m)*g+m*l*dq^2*cos(q)+Fy; ...
        Fp; ...
        m*g*l*sin(q)+Fq];
  rhs = A\bb;
  dX(1) = X(2); dX(2) = rhs(1);
  dX(3) = X(4); dX(4) = rhs(2);
  dX(5) = X(6); dX(6) = rhs(3);
  dX(7) = X(8); dX(8) = rhs(4);
endfunction
n=400; t=linspace(0,25,n);
x0=[0; 0; 2*r; 0; ...
    0; 0; %pi/6; 0];
```

```
xx=ode( x0, 0, t, eom );
//アニメーションする.
function draw_wheel(x,y,r,ph,l,th)
 // x, y, 車輪半径, 車輪角, 振子長, 振子角
 g=gca();                    //座標軸の取得
 g.data_bounds=[-5,-1;5,1]; //座標軸の範囲
 g.isoview="on";             //縦横比1; グリッド;
 xset("color",1);            //基本色黒
// xsegs(5*[-1;1],-r*[1;1],9); //地面青
 xsegs(5*[-1;1],0*[1;1],9); //地面青
 xarc(x-r, y+r, 2*r, 2*r, 0, 360*64); //車輪
 a=ph+2*%pi*(0:4)/5;
 xx=x*ones(a); yy=y*ones(a);
 xsegs([xx;xx+r*sin(a)],[yy;yy+r*cos(a)]); //スポーク
 xsegs([x;x+l*sin(th)],[y;y+l*cos(th)],5); //振り子
 p=gce(); p.thickness=2;       //直前の描画; 線の太さ
endfunction
x=x0(1); y=x0(3); p=x0(5); q=x0(7);
draw_wheel(x,y,r,p,l,q); drawnow;  //描画実行; 画面更新;
xclick(); //マウスクリック待ち
realtimeinit(0.05); //アニメーションの時間刻み
for i=1:n
 realtime(i);
 drawlater();        //描画延期
 x=xx(1,i); y=xx(3,i); p=xx(5,i); q=xx(7,i);
 clf();
 draw_wheel(x,y,r,p,l,q);
 drawnow;            //画面更新;
end
```