```
//--------------------------------------------------------------------------------------------
/*! ¥file
    ¥brief  4-legged robot simulator - client
    ¥author Akihiko Yamaguchi
    ¥date   Mar.13 2007  */
//--------------------------------------------------------------------------------------------
#include <iostream> // TODO デバッグが終了しだい削除
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
//--------------------------------------------------------------------------------------------
#include "protocol.h"
//--------------------------------------------------------------------------------------------
#include <octave/config.h>
#include <octave/Matrix.h>
//--------------------------------------------------------------------------------------------
#ifdef OUTPUT_OCT
#include <octave/oct.h>
#endif
//--------------------------------------------------------------------------------------------
static int  client_file_descriptor(-1);
static int  JOINT_NUM(0);
static int  JOINT_STATE_DIM (0);  // 関節状態の次元
static int  BASE_STATE_DIM (0);  // ベース状態の次元
//--------------------------------------------------------------------------------------------
static double *joint_state (NULL);
static double *base_state (NULL);
static ColumnVector jState(0), bState(0);
//--------------------------------------------------------------------------------------------
class __inner_destructor
{
  __inner_destructor(void) {};
  ~__inner_destructor(void)
    {
      if (joint_state!=NULL) {delete[] joint_state; joint_state=NULL;}
      if (base_state!=NULL)  {delete[] base_state; base_state=NULL;}
    };
};
//--------------------------------------------------------------------------------------------


inline ColumnVector get_joint_state (void);
inline ColumnVector get_base_state (void);

using namespace std;

//! check return
void chret (int ret)
{
  if (ret<0)
  {
    close (client_file_descriptor);
    client_file_descriptor = -1;
    exit(1);
  }
}

//! check the client_file_descriptor
void chfd (void)
{
  if(client_file_descriptor<0)
  {
    cerr<<"error! the connection was already terminated."<<endl;
    exit(1);
  }
}

bool setup_client (void)
  // ref. http://www.ueda.info.waseda.ac.jp/~toyama/network/example1.html
{
  struct sockaddr_un    addr;

  // ソケットを作成.  UNIX ドメイン, ストリーム型
  if ((client_file_descriptor = socket (PF_UNIX, SOCK_STREAM, 0)) < 0)
  {
    perror("socket");
    exit(1);
  }

  bzero ((char *)&addr, sizeof(addr));

  // ソケットの名前を代入
  addr.sun_family = AF_UNIX;
  strcpy (addr.sun_path, SOCK_NAME);

  // サーバと接続を試みる. サーバ側で bind & listen の発行が終っている必要がある
  if (connect (client_file_descriptor, (struct sockaddr *)&addr,
       sizeof(addr.sun_family) + strlen(SOCK_NAME)) < 0)
  {
    perror("connect");
    cerr<<"-> maybe the server four-legged.exe is not running."<<endl;
    return false; //exit(1);
  }
  return true;
}

ColumnVector get_torque (void)
{
  static bool init(true);
  static const double kp=100.0, kd=2.0;
  static ColumnVector target(JOINT_NUM,0.0);
```

```
106   static const double MaxTorque(100.0);  // [Nm]
107   if(init)
108   {
109     const double q1=-0.25*M_PI, q2=0.5*M_PI;
110     for(int i(0);i<8;i+=2) target(i)=q1;
111     for(int i(1);i<8;i+=2) target(i)=q2;
112     init = false;
113   }
114   ColumnVector u (JOINT_NUM,0.0);  // 制御入力（トルク）
115   ColumnVector jstate (get_joint_state());  // 現在の関節状態
116   for (int i(0);i<8;++i)
117   {
118     u(i)=kp*(target(i)-jstate(i))-kd*jstate(8+i);
119     if (u(i)>MaxTorque)  u(i)=MaxTorque;
120     else if (u(i)<-MaxTorque)  u(i)=-MaxTorque;
121   }
122
123   return u;
124 }
125 //-----------------------------------------------------------------------------------------
126
127 inline void start_simulation (int window_width, int window_height)
128 {
129   chfd();
130   TXData data;
131   data.command = ORS_SET_WINDOWSIZE;
132   data.step = 0;
133   data.ivalue = window_width;
134   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
135   data.step = 1;
136   data.ivalue = window_height;
137   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
138   data.command = ORS_START_SIM;
139   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
140 }
141
142 inline void stop_simulation (void)
143 {
144   chfd();
145   TXData data;
146   data.command = ORS_STOP_SIM;
147   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
148   close(client_file_descriptor);
149   client_file_descriptor=-1;
150 }
151
152 inline void step_simulation (const ColumnVector &u, const double &time_step)
153 {
154   chfd();
155   TXData data;
156   data.command = ORS_SET_TORQUE;
157   for (int j(0); j<JOINT_NUM; ++j)
158   {
159     data.step = j;
160     data.dvalue = u(j);
161     chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
162   }
163   data.command = ORS_STEP_SIM;
164   data.dvalue = time_step;
165   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
166 }
167
168 inline void reset_simulation (void)
169 {
170   chfd();
171   TXData data;
172   data.command = ORS_RESET_SIM;
173   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
174 }
175
176 inline ColumnVector get_joint_state (void)
177 {
178   chfd();
179   if (JOINT_STATE_DIM<=0)  return ColumnVector(0);
180   if (jState.dim1() != JOINT_STATE_DIM)  jState.resize(JOINT_STATE_DIM);
181   TXData data;
182   data.command = ORS_GET_JOINT_STATE;
183   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
184   chret (read (client_file_descriptor, (char*)joint_state, sizeof(double)*JOINT_STATE_DIM));
185   // cerr<<"c-joint1= ";for(int j(0);j<JOINT_STATE_DIM;++j)cerr<<" "<<joint_state[j];cerr<<endl;
186   for (int j(0); j<JOINT_STATE_DIM; ++j)  jState(j)=joint_state[j];
187   // cerr<<"c-joint2= "<<jState.transpose()<<endl;
188   return jState;
189 }
190 inline ColumnVector get_base_state (void)
191 {
192   chfd();
193   if (BASE_STATE_DIM<=0)  return ColumnVector(0);
194   if (bState.dim1() != BASE_STATE_DIM)  bState.resize(BASE_STATE_DIM);
195   TXData data;
196   data.command = ORS_GET_BASE_STATE;
197   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
198   chret (read (client_file_descriptor, (char*)base_state, sizeof(double)*BASE_STATE_DIM));
199   for (int j(0); j<BASE_STATE_DIM; ++j)  bState(j)=base_state[j];
200   return bState;
201 }
202
203 inline void draw_world (void)
204 {
205   chfd();
206   TXData data;
207   data.command = ORS_DRAW_WORLD;
208   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
209 }
210
```

```
211 inline int get_joint_num (void)
212 {
213   chfd();
214   TXData data;
215   data.command = ORS_GET_JOINT_NUM;
216   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
217   chret (read (client_file_descriptor, (char*)&JOINT_NUM, sizeof(JOINT_NUM)));
218   cerr<<"joint-num = "<<JOINT_NUM<<endl;
219   return JOINT_NUM;
220 }
221
222 inline int get_joint_state_dim (void)
223 {
224   chfd();
225   TXData data;
226   data.command = ORS_GET_JSTATE_DIM;
227   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
228   chret (read (client_file_descriptor, (char*)&JOINT_STATE_DIM, sizeof(JOINT_STATE_DIM)));
229   cerr<<"joint-state-dim = "<<JOINT_STATE_DIM<<endl;
230   if (joint_state!=NULL) {delete[] joint_state; joint_state=NULL;}
231   joint_state = new double[JOINT_STATE_DIM];
232   return JOINT_STATE_DIM;
233 }
234
235 inline int get_base_state_dim (void)
236 {
237   chfd();
238   TXData data;
239   data.command = ORS_GET_BSTATE_DIM;
240   chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
241   chret (read (client_file_descriptor, (char*)&BASE_STATE_DIM, sizeof(BASE_STATE_DIM)));
242   cerr<<"base-state-dim = "<<BASE_STATE_DIM<<endl;
243   if (base_state!=NULL) {delete[] base_state; base_state=NULL;}
244   base_state = new double[BASE_STATE_DIM];
245   return BASE_STATE_DIM;
246 }
247
248 #ifdef OUTPUT_OCT
249 DEFUN_DLD (startSimulation, args, ,
250   "int startSimulation(int window_width, int window_height).")
251 {
252   if(!setup_client()) return octave_value(1);
253   start_simulation(args(0).double_value(), args(1).double_value());
254   get_joint_num();
255   get_joint_state_dim();
256   get_base_state_dim();
257   return octave_value(0);
258 }
259
260 DEFUN_DLD (stopSimulation, args, ,
261   "void stopSimulation(void).")
262 {
263   stop_simulation();
264   return octave_value();
265 }
266
267 DEFUN_DLD (stepSimulation, args, ,
268   "void stepSimulation(const ColumnVector &u, const dReal &time_step).")
269 {
270   ColumnVector u(args(0).vector_value());
271   step_simulation (u, args(1).double_value());
272   return octave_value();
273 }
274
275 DEFUN_DLD (resetSimulation, args, ,
276   "void resetSimulation(void).")
277 {
278   reset_simulation();
279   return octave_value();
280 }
281
282 DEFUN_DLD (drawWorld, args, ,
283   "void drawWorld(void).")
284 {
285   draw_world();
286   return octave_value();
287 }
288
289 DEFUN_DLD (getJointState, args, ,
290   "ColumnVector getJointState(void).")
291 {
292   ColumnVector state (get_joint_state());
293   return octave_value (state);
294 }
295
296 DEFUN_DLD (getBaseState, args, ,
297   "ColumnVector getBaseState(void).")
298 {
299   ColumnVector state (get_base_state());
300   return octave_value (state);
301 }
302 #endif
303
304
305 int main (int argc, char **argv)
306 {
307   if(!setup_client()) return 1;
308   start_simulation(400,400);
309   get_joint_num();
310   get_joint_state_dim();
311   get_base_state_dim();
312   while(1)
313   {
314     step_simulation (get_torque(), 0.001);
315     draw_world();
```

```
316   }
317
318   close (client_file_descriptor);
319   return 0;
320 }
321 //--------------------------------------------------------------------------------
322
323
```

```
//------------------------------------------------------------------------------------------------
/*! ¥file
    ¥brief  4-legged robot simulator - server
    ¥author Akihiko Yamaguchi
    ¥date   Mar.13 2007  */
//------------------------------------------------------------------------------------------------
#ifndef ODE_MINOR_VERSION
  #error ODE_MINOR_VERSION should be set in compile
  #error   ex. -DODE_MINOR_VERSION=10
#endif
//------------------------------------------------------------------------------------------------
#include <ode/ode.h>
#include <drawstuff/drawstuff.h>
#include <iostream>
#undef PACKAGE_BUGREPORT
#undef PACKAGE_NAME
#undef PACKAGE_STRING
#undef PACKAGE_TARNAME
#undef PACKAGE_VERSION
#include <octave/config.h>
#include <octave/Matrix.h>
//------------------------------------------------------------------------------------------------
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
//------------------------------------------------------------------------------------------------
#include "protocol.h"
//------------------------------------------------------------------------------------------------
#ifdef _MSC_VER
#pragma warning(disable:4244 4305)  // for VC++, no precision loss complaints
#endif
// select correct drawing functions
#ifdef dDOUBLE
#define dsDrawBox dsDrawBoxD
#define dsDrawSphere dsDrawSphereD
#define dsDrawCylinder dsDrawCylinderD
#define dsDrawCapsule dsDrawCapsuleD
#define dsDrawConvex dsDrawConvexD
#endif
//------------------------------------------------------------------------------------------------
using namespace std;
//------------------------------------------------------------------------------------------------
#include "robot.cpp"
//------------------------------------------------------------------------------------------------


//================================================================================================
//! ¥brief ふたつのオブジェクト o1, o2 が衝突しそうならこのコールバック関数が呼ばれる
//! ¥note 衝突しているかいないかはこの関数で（ユーザが）判定し，衝突していれば接触点にリンクを追加する.
static void nearCallback (void *data, dGeomID o1, dGeomID o2)
//================================================================================================
{
  // exit without doing anything if the two bodies are connected by a joint
  dBodyID b1 = dGeomGetBody(o1);
  dBodyID b2 = dGeomGetBody(o2);
  if (b1 && b2 && dAreConnectedExcluding(b1,b2,dJointTypeContact)) return;

  dContact contact[MAX_CONTACTS];   // up to MAX_CONTACTS contacts per box-box
  for (int i=0; i<MAX_CONTACTS; i++)
  {
    contact[i].surface.mode = dContactBounce | dContactSoftCFM;
    contact[i].surface.mu = dInfinity;
    contact[i].surface.mu2 = 0;
    contact[i].surface.bounce = 0.1;
    contact[i].surface.bounce_vel = 0.1;
    contact[i].surface.soft_cfm = 0.01;
  }
  if (int numc = dCollide (o1,o2,MAX_CONTACTS,&contact[0].geom,sizeof(dContact)))
  {
    for (int i=0; i<numc; i++)
    {
      dJointID c = dJointCreateContact (world.id(),contactgroup.id(),contact+i);
      dJointAttach (c,b1,b2);
    }
  }
}
//------------------------------------------------------------------------------------------------


//================================================================================================
//! ¥brief start simulation - set viewpoint
static void start()
//================================================================================================
{
  #if ODE_MINOR_VERSION>=10
    dAllocateODEDataForThread(dAllocateMaskAll);
  #endif

  static float xyz[3] = {0.75,1.3,1.0};
  static float hpr[3] = {-120.0,-16.0,0.0};

  dsSetViewpoint (xyz,hpr);
  // cerr << "Press 'R' to reset simulation¥n" << endl;
}
//------------------------------------------------------------------------------------------------

//================================================================================================
//! ¥brief キーイベントのコールバック関数
//! ¥param[in] cmd  入力キー
static void keyEvent (int cmd)
//================================================================================================
```

```
106 {
107   // if (cmd=='r'||cmd=='R')
108   // {
109   //   create_world();
110   // }
111 }
112 //-------------------------------------------------------------------------------------------------
113
114 static void getJointState (double state[JOINT_STATE_DIM])
115 {
116   for (int j(0);j<JOINT_NUM;++j)
117   {
118     state[j] = joint[j].getAngle();
119     state[JOINT_NUM+j] = joint[j].getAngleRate();
120   }
121   // cerr<<"joint1= ";for(int j(0);j<JOINT_STATE_DIM;++j)cerr<<" "<<state[j];cerr<<endl;
122 }
123 //-------------------------------------------------------------------------------------------------
124
125 static void getBaseState (double state[BASE_STATE_DIM])
126 {
127   state[0]  = body[0].getPosition()[0];  // x
128   state[1]  = body[0].getPosition()[1];  // y
129   state[2]  = body[0].getPosition()[2];  // z
130   state[3]  = body[0].getQuaternion()[0]; // quaternion (w)
131   state[4]  = body[0].getQuaternion()[1]; // quaternion (x)
132   state[5]  = body[0].getQuaternion()[2]; // quaternion (y)
133   state[6]  = body[0].getQuaternion()[3]; // quaternion (z)
134
135   state[7]  = body[0].getLinearVel()[0];  // vx
136   state[8]  = body[0].getLinearVel()[1];  // vy
137   state[9]  = body[0].getLinearVel()[2];  // vz
138   state[10] = body[0].getAngularVel()[0] ; // wx
139   state[11] = body[0].getAngularVel()[1] ; // wx
140   state[12] = body[0].getAngularVel()[2] ; // wx
141 }
142 //-------------------------------------------------------------------------------------------------
143
144
145
146 //-------------------------------------------------------------------------------------------------
147 static int    global_file_descriptor(-1);
148 static int    window_x(400), window_y(400);
149 // static const  dReal time_step (0.0005); // シミュレーションきざみ幅(0.5[ms])
150 static ColumnVector  input_torque (JOINT_NUM, 0.0);
151 //-------------------------------------------------------------------------------------------------
152
153
154
155 void stepSimulation (const dReal &time_step)
156 {
157   for (int j(0); j<JOINT_NUM; ++j)
158     joint[j].addTorque(input_torque(j));
159   // cerr<<"torque= "<<input_torque.transpose()<<endl;
160
161   // シミュレーション
162   space.collide (0,&nearCallback);
163   world.step (time_step);
164   // time += time_step;
165   // remove all contact joints
166   contactgroup.empty();
167 }
168 //-------------------------------------------------------------------------------------------------
169
170
171 bool oct_robot_server (void)
172 {
173   TXData data;
174   while (1)
175   {
176     if (global_file_descriptor<0)
177     {
178       cerr<<"connection terminated (unexpected error)."<<data.command<<endl;
179       exit(1);
180     }
181     read (global_file_descriptor, (char*)&data, sizeof(data));
182     switch (data.command)
183     {
184       case  ORS_START_SIM        :
185         return true;
186       case  ORS_STOP_SIM         :
187         return false;
188       case  ORS_STEP_SIM         :
189         stepSimulation (data.dvalue);
190         break;
191       case  ORS_RESET_SIM        :
192         create_world();
193         break;
194       case  ORS_DRAW_WORLD       :
195         return true;
196       case  ORS_SET_TORQUE       :
197         input_torque(data.step) = data.dvalue;
198         break;
199       case  ORS_SET_WINDOWSIZE   :
200         if (data.step==0)  window_x=data.ivalue;
201         else if (data.step==1)  window_y=data.ivalue;
202         break;
203       case  ORS_GET_JOINT_NUM    :
204         write (global_file_descriptor, (char*)&JOINT_NUM, sizeof(JOINT_NUM));
205         break;
206       case  ORS_GET_JSTATE_DIM   :
207         write (global_file_descriptor, (char*)&JOINT_STATE_DIM, sizeof(JOINT_STATE_DIM));
208         break;
209       case  ORS_GET_BSTATE_DIM   :
210         write (global_file_descriptor, (char*)&BASE_STATE_DIM, sizeof(BASE_STATE_DIM));
```

```
211         break;
212       case  ORS_GET_JOINT_STATE  :
213         getJointState (joint_state);
214         // cerr<<"joint2= ";for(int j(0);j<JOINT_STATE_DIM;++j)cerr<<" "<<joint_state[j];cerr<<endl;
215         write (global_file_descriptor, (char*)joint_state, sizeof(double)*JOINT_STATE_DIM);
216         break;
217       case  ORS_GET_BASE_STATE   :
218         getBaseState (base_state);
219         write (global_file_descriptor, (char*)base_state, sizeof(double)*BASE_STATE_DIM);
220         break;
221       default :
222         cerr<<"in oct_robot_server(): invalid command "<<data.command<<endl;
223         return false;
224     }
225   }
226 }
227 //-------------------------------------------------------------------------------------------
228
229 void setup_server (void)
230   // ref. http://www.ueda.info.waseda.ac.jp/~toyama/network/example1.html
231 {
232   int    fd1;
233   struct sockaddr_un    saddr;
234   struct sockaddr_un    caddr;
235   int    len;
236
237   if ((fd1 = socket (PF_UNIX, SOCK_STREAM, 0)) < 0)
238   {
239     perror("socket");
240     exit(1);
241   }
242
243   bzero ((char *)&saddr, sizeof(saddr));
244   // ソケットの名前を代入
245   saddr.sun_family = AF_UNIX;
246   strcpy (saddr.sun_path, SOCK_NAME);
247   // ソケットにアドレスをバインド
248   unlink(SOCK_NAME);
249   if (bind(fd1, (struct sockaddr *)&saddr,
250           sizeof(saddr.sun_family) + strlen(SOCK_NAME)) < 0){
251     perror("bind");
252     exit(1);
253   }
254   // listen をソケットに対して発行
255   if (listen(fd1, 1) < 0)
256   {
257     perror("listen");
258     exit(1);
259   }
260
261   len = sizeof(caddr);
262   /*
263     * accept()により、クライアントからの接続要求を受け付ける。
264     * 成功すると、クライアントと接続されたソケットのディスクリプタが
265     * fd2に返される。このfd2を通して通信が可能となる。
266     * fd1は必要なくなるので、close()で閉じる。
267     */
268   if ((global_file_descriptor = accept(fd1, (struct sockaddr *)&caddr, (socklen_t*)&len)) < 0)
269   {
270       perror("accept");
271       exit(1);
272   }
273   close(fd1);
274 }
275 //-------------------------------------------------------------------------------------------
276
277
278 //===========================================================================================
279 /*! ¥brief 描画(OpenGL)のコールバック関数.
280     ¥param[in] pause 停止モードなら true (0以外)
281
282     シミュレーションのきざみ time_step=0.0005[s] に対して描画は 50 fps 程度で十分なので,
283     1 frame ごとに simStepsPerFrame=1.0/time_step/FPS=40 回ダイナミクスのシミュレーションを回す.  */
284 static void simLoop (int pause)
285 //===========================================================================================
286 {
287 //   static dReal time(0.0); // シミュレーション時間
288   if (!pause)
289   {
290     if (!oct_robot_server())  dsStop();
291   }
292
293   draw_world();
294 }
295 //-------------------------------------------------------------------------------------------
296
297 static void stopSimulation (void)
298 {
299   close (global_file_descriptor);
300   global_file_descriptor = -1;
301 }
302 //-------------------------------------------------------------------------------------------
303
304
305 int main (int argc, char **argv)
306 {
307   dsFunctions fn; // OpenGL 出力用オブジェクト
308   fn.version = DS_VERSION;
309   fn.start = &start;
310   fn.step = &simLoop;
311   fn.command = &keyEvent;
312   fn.stop = &stopSimulation;
313   char path_to_textures[]="textures";
314   fn.path_to_textures = path_to_textures;  //! ¥note カレントディレクトリに textures へのリンクが必要
315
```

```
316   #if ODE_MINOR_VERSION>=9
317   # if ODE_MINOR_VERSION==9
318     dInitODE();
319   # elif ODE_MINOR_VERSION>=10
320     dInitODE2(0);
321   # endif
322   #endif
323
324   setup_server();
325   if (oct_robot_server())
326   {
327     create_world();
328     // run simulation
329     if(window_x>0 && window_y>0)
330       dsSimulationLoop (argc,argv,window_x,window_y,&fn);
331     else
332       while(oct_robot_server());
333   }
334   if(global_file_descriptor!=-1)
335   {
336     close (global_file_descriptor);
337     global_file_descriptor = -1;
338   }
339
340   #if ODE_MINOR_VERSION>=9
341     dCloseODE();
342   #endif
343   return 0;
344 }
345 //-----------------------------------------------------------------------------------------
346
```

```
function [sigma, mu]=NaturalActorCritic(L, M, T, options, winx,winy)
    startSimulation (winx,winy);    % 本体のウィンドウを表示
    MaxTorque = 100;            % 最大トルク
    MinTorque = -100;           % 最小トルク
    N = 19;             % モデルパラメータ数 (mu:18次元, sigma:1次元)
    ibstate = getBaseState();       % 胴体の初期状態

    % 政策モデルパラメータをランダムに初期化
    mu = rand(N-1,1)-0.5;
    sigma = rand*10;

    % デザイン行列Z, 報酬ベクトルqおよび
    % アドバンテージ関数のモデルパラメータwの初期化
    Z = zeros(M,N);
    q = zeros(M,1);
    w = zeros(N,1);

    % 政策反復
    for l=1:L
        dr = 0;
        rand('state',l);

        % 標本
        for m=1:M
        resetSimulation();

        for t=1:T
        % 状態の初期化
            state = zeros(N-2,1);

        % 関節状態観測
                jstate = getJointState();
                bstate = getBaseState();

        % 状態ベクトルの構築
                state(1:16) = jstate;  % 8関節の角度および速度
                state(17) = bstate(3); % 胴体z軸方向の位置
                state(18) = bstate(10);% 胴体z軸方向の速度

        % 行動の選択
        action = randn*sigma + mu'*state;
        action = min(action,MaxTorque);% 最小値確認
        action = max(action,MinTorque);% 最大値確認

        % 行動の実行
        u = zeros(1,8);
        u(2) = action;
        u(4) = action;
        u(6) = action;
        u(8) = action;

                stepSimulation (u,0.0005);
                if(t==0 || mod(t,50)==0)
                   drawWorld;
                end
```

```
        % 胴体状態観測
                abstate = getBaseState();

        % 状態, 行動および報酬のデータを記録
                states(:,t)  = state;
                actions(t)   = action;
                rewards(m,t) = abstate(1) - ibstate(1);
        dr   = dr + options.gamma^(t-1)*rewards(m,t);
            end

            for t=1:T
        % 平均muに関する勾配の計算
        der(1:N-1) = (actions(t)-mu'*states(:,t))*states(:,t)/(sigma^2);

        % 標準偏差sigmaに関する勾配の計算
            der(N) = ((actions(t)-mu'*states(:,t))^2-sigma^2)/(sigma^3);

        % デザイン行列Zおよびqベクトル
        Z(m,:) = Z(m,:) + options.gamma^(t-1)*der;
        q(m) = q(m) + options.gamma^(t-1)*(rewards(m,t));
        end
        end

    % r - V(s1)
        q = q - dr/M;

        % 最小二乗法を用いてアドバンテージ関数のモデルパラメータwを推定
    Z(:,N) = ones(M,1);
    w = pinv(Z'*Z)*Z'*q;

    % wを用いてモデルパラメータを更新
    mu = mu + options.alpha*w(1:N-1);
    sigma = sigma + options.alpha*w(N);

        printf("%d) Max=%.2f Min=%.2f Avg=%.2f Dsum=%.2f¥n",l,max(max(rewards)), min(min(rewards))↙
mean(mean(rewards)), dr/M);
        fflush(stdout);
    end
```

```
function [sigma, mu]=PolicyGradient(L, M, T, options, winx,winy)
  startSimulation (winx,winy); % 本体のウィンドウを表示
  MaxTorque = 100;      % 最大トルク
  MinTorque = -100;      % 最小トルク
  N = 19;        % モデルパラメータ数 (mu:18次元, sigma:1次元)
  ibstate = getBaseState();      % 胴体の初期状態

  % 政策モデルパラメータをランダムに初期化
  mu = rand(N-1,1)-0.5;
  sigma = rand*10;

  % 政策反復
  for l=1:L
    dr = 0;
    rand('state',l);

    % 標本
    for m=1:M
      drs(m) = 0;
      der(m,:) = zeros(1,N);

      resetSimulation();

      for t=1:T
        % 状態の初期化
        state = zeros(N-2,1);

        % 関節状態観測
        jstate = getJointState();
        bstate = getBaseState();

        % 状態ベクトルの構築
        state(1:16) = jstate;   % 8関節の角度および速度
        state(17) = bstate(3);  % 胴体z軸方向の位置
        state(18) = bstate(10);% 胴体z軸方向の速度

        % 行動の選択
        action = randn*sigma + mu'*state;
        action = min(action,MaxTorque);% 最小値確認
        action = max(action,MinTorque);% 最大値確認

        % 行動の実行
        u = zeros(1,8);
        u(2) = action;
        u(4) = action;
        u(6) = action;
        u(8) = action;

        stepSimulation (u,0.0005);
        if(t==0 || mod(t,50)==0)
           drawWorld;
        end

        % 胴体状態観測
        abstate = getBaseState();
```

```
        % 平均muに関する勾配の計算
        der(m,1:N-1) = der(m,1:N-1) + ((action-mu'*state)*state/(sigma^2))';

        % 標準偏差sigmaに関する勾配の計算
        der(m,N) = der(m,N) + ((action-mu'*state)^2-sigma^2)/(sigma^3);

        % 割引き報酬和の計算
        rewards(m,t) = abstate(1) - ibstate(1);
        drs(m) = drs(m) + options.gamma^(t-1)*rewards(m,t);
        dr = dr + options.gamma^(t-1)*rewards(m,t);
      end
    end

    % 最少分散ベースラインを計算
    b = drs * diag(der*der') / trace(der*der');

    % 勾配を推定
    derJ = 1/M * ((drs-b) * der)';

    % モデルパラメータを更新
    mu = mu + options.alpha * derJ(1:N-1);
    sigma = sigma + options.alpha * derJ(N);

    printf("%d) Max=%.2f Min=%.2f Avg=%.2f Dsum=%.2f¥n",l,max(max(rewards)), min(min(rewards)), mean
(mean(rewards)), dr/M);
    fflush(stdout);
  end
```

```
//--------------------------------------------------------------------------------
/*! ¥file
    ¥brief  create 4-legged robot for ODE
    ¥author Akihiko Yamaguchi
    ¥date   Mar.20 2007 */
//--------------------------------------------------------------------------------
// dynamics and collision objects
static dWorld world;
static dSimpleSpace space (0);
static dPlane plane;
static dBody body[9];
static const int JOINT_NUM(8);
static const int JOINT_STATE_DIM(JOINT_NUM*2);  // 関節状態の次元
static const int BASE_STATE_DIM(13);  // ベース状態の次元
static dHingeJoint joint[JOINT_NUM];
static dJointGroup contactgroup;
static dBox  LinkTorso;
static dCapsule LinkLeg[8];

static const int  MAX_CONTACTS (10);  // maximum number of contact points per body
//--------------------------------------------------------------------------------
static double joint_state[JOINT_STATE_DIM];
static double base_state[BASE_STATE_DIM];
//--------------------------------------------------------------------------------
static const dReal _scale       = 0.5;
static const dReal param_h0     = 0.1    * _scale;
static const dReal param_wx0    = 1.6    * _scale;
static const dReal param_wy0    = 0.8    * _scale;
static const dReal param_px     = 0.14   * _scale;
static const dReal param_py     = 0.10   * _scale;
static const dReal param_d1     = 0.15   * _scale;
static const dReal param_l1     = 0.5    * _scale;
static const dReal param_d2     = 0.15   * _scale;
static const dReal param_l2     = 0.5    * _scale;
static const dReal param_dj     = 0.25   * _scale;

static const dReal density      = 2000.0;  // 各リンクの密度[kg/m^3]. 参考(?)`人体の密度' は 900~1100 kg/m^3 (wikipedia)
//--------------------------------------------------------------------------------


//================================================================================
//! ¥brief シミュレーションオブジェクトを作成
void create_world (void)
//================================================================================
{
  int j;
  contactgroup.create (0);
  world.setGravity (0,0,-9.8);  // 重力 [m/s^2]
  dWorldSetCFM (world.id(),1e-5);
  plane.create (space,0,0,1,0); // 地面（平面）.

  const dReal cx=0.0, cy=0.0, cz=param_l1+param_l2;
  j=0; { // 胴体
    body[j].create (world);
    body[j].setPosition (cx, cy, cz);
    dReal xx=param_wx0, yy=param_wy0, zz=param_h0;
    dMass m;
    m.setBox (density,xx,yy,zz);
    body[j].setMass (&m);
    LinkTorso.create (space,xx,yy,zz);
    LinkTorso.setBody (body[j]);
  }
  for (int k(0); k<4; ++k)
  {
    dReal xx, yy, zz;
    if (k==0 || k==1)  xx=cx+0.5*param_wx0-param_px;
    else               xx=cx-0.5*param_wx0+param_px;
    if (k==0 || k==2)  yy=cy+0.5*param_wy0+param_py;
    else               yy=cy-0.5*param_wy0-param_py;
    // 脚
    for (int i(0); i<2; ++i)
    {
      j=2*k+i+1;
      dReal rad, len;
      if (i==0) {rad=0.5*param_d1; len=param_l1-2.0*rad; zz=cz-0.5*param_l1;}
      else      {rad=0.5*param_d2; len=param_l2-2.0*rad; zz=cz-param_l1-0.5*param_l2;}
      body[j].create (world);
      body[j].setPosition (xx, yy, zz); // リンク1の中心座標
      dMass m;
      m.setCapsule (density,3,rad,len);  // direction(3): z-axis
      body[j].setMass (&m);
      LinkLeg[j-1].create (space,rad,len);
      LinkLeg[j-1].setBody (body[j]);
    }
    // 関節
    dBodyID b1, b2;
    for (int i(0); i<2; ++i)
    {
      j=2*k+i;
      if (i==0)  {b1=body[0]; b2=body[j+1]; zz=cz;}
      else       {b1=body[j]; b2=body[j+1]; zz=cz-param_l1;}
      joint[j].create (world);
      joint[j].attach (b1,b2);
      joint[j].setAnchor (xx,yy,zz); // 回転中心=支柱の中心(=原点)
      joint[j].setAxis (0.0,1.0,0.0); // 回転軸=y軸
      joint[j].setParam (dParamHiStop, +0.5*M_PI); // 関節の可動範囲を制約するときに使う
      joint[j].setParam (dParamLoStop, -0.5*M_PI); // acrobot の場合は省略
    }
  }
}
//--------------------------------------------------------------------------------


//================================================================================
//! ¥brief 描画関数
```

```
106 void draw_world( void )
107 //=================================================================================================
108 {
109   int j;
110   dsSetColor (0,0.5,1);
111   dsSetTexture (DS_WOOD);
112   dReal rad, len;
113   dReal sides[4];
114   dVector3 pos;
115   dBox *blink;
116   dCapsule *clink;
117   dsSetTexture (DS_NONE);
118   dsSetColorAlpha (1.0, 1.0, 1.0, 0.8);
119   j=0; blink=&LinkTorso; blink->getLengths(sides); dsDrawBox (blink->getPosition(),blink->getRotation(),sides);
120   for (j=1; j<=8; ++j)
121     {clink=&LinkLeg[j-1]; clink->getParams(&rad, &len); dsDrawCapsule (clink->getPosition(),clink->getRotation(),len,rad);}
122   dsSetColorAlpha (0.0, 1.0, 0.0, 0.6);
123   for (j=0; j<8; ++j)
124     {joint[j].getAnchor(pos); dsDrawSphere (pos, body[0].getRotation(), 0.5*param_dj);}
125 }
126 //-------------------------------------------------------------------------------------------------
127
128
129
130
```

```
function test()
  if(startSimulation(600,400)~=0) return; endif % シミュレーション開始(ウィンドウサイズ)
  %if(startSimulation(0,0)~=0) return; endif % シミュレーション開始(ウィンドウを表示しない)
  more off
  printf('press any key..¥n')
  kbhit();
  printf('press a,s,d,f,g: change angles¥n')
  printf('press w,b: show state¥n')
  printf('press r: reset¥n')
  printf('press q: quit¥n')
  kp=80; kd=2;
  MaxTorque = 100.0;
  q1=0.25*pi; q2=-0.5*pi;
  q3=0.125*pi; q4=-0.25*pi;
  target = [q3,q4,q3,q4,q3,q4,q3,q4]';
  ii=0;
  while (1)
    jstate = getJointState();% robot の関節状態 [q1,..,q8,dq1,..,dq8]' を返す
    u = zeros(8,1);
    for i=1:8
      u(i)=kp*(target(i)-jstate(i))-kd*jstate(8+i);
      if (u(i) > MaxTorque)  u(i) = MaxTorque;  endif
      if (u(i) < -MaxTorque) u(i) = -MaxTorque; endif
    endfor
    stepSimulation (u,0.0005); % (トルク, 時間幅)でシミュレーションを進める
    if (ii==0)   drawWorld();  endif % 25 回に 1 回の割合で描画 (描画が重いことを想定して)
    ii++;
    if (ii==25) ii=0; endif
    key=kbhit(1);
    switch key
      case 'q';  stopSimulation(); printf(¥n'); return;    % シミュレーションを終了
      case 'a';  target = [q1,q2,q1,q2,q1,q2,q1,q2]';
      case 's';  target = [q3,q4,q3,q4,q3,q4,q3,q4]';
      case 'd';  target = zeros(8,1);
      case 'f';  target = -[q3,q4,q3,q4,q3,q4,q3,q4]';
      case 'g';  target = -[q1,q2,q1,q2,q1,q2,q1,q2]';
      case 'w';  getJointState()
      case 'b';  getBaseState()
      case 'r';  resetSimulation();
    endswitch
  endwhile
```