```

//-------------------------------------------------------------------------------------------
/*! ¥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)
```

```
//=============================================================================================
{
  // if (cmd=='r'||cmd=='R')
  // {
  //   create_world();
  // }
}
//---------------------------------------------------------------------------------------------

static void getJointState (double state[JOINT_STATE_DIM])
{
  for (int j(0);j<JOINT_NUM;++j)
  {
    state[j] = joint[j].getAngle();
    state[JOINT_NUM+j] = joint[j].getAngleRate();
  }
  // cerr<<"joint1= ";for(int j(0);j<JOINT_STATE_DIM;++j)cerr<<" "<<state[j];cerr<<endl;
}
//---------------------------------------------------------------------------------------------

static void getBaseState (double state[BASE_STATE_DIM])
{
  state[0]  = body[0].getPosition()[0];  // x
  state[1]  = body[0].getPosition()[1];  // y
  state[2]  = body[0].getPosition()[2];  // z
  state[3]  = body[0].getQuaternion()[0]; // quaternion (w)
  state[4]  = body[0].getQuaternion()[1]; // quaternion (x)
  state[5]  = body[0].getQuaternion()[2]; // quaternion (y)
  state[6]  = body[0].getQuaternion()[3]; // quaternion (z)

  state[7]  = body[0].getLinearVel()[0];  // vx
  state[8]  = body[0].getLinearVel()[1];  // vy
  state[9]  = body[0].getLinearVel()[2];  // vz
  state[10] = body[0].getAngularVel()[0] ; // wx
  state[11] = body[0].getAngularVel()[1] ; // wx
  state[12] = body[0].getAngularVel()[2] ; // wx
}
//---------------------------------------------------------------------------------------------



//---------------------------------------------------------------------------------------------
static int    global_file_descriptor(-1);
static int    window_x(400), window_y(400);
// static const  dReal time_step (0.0005); // シミュレーションきざみ幅(0.5[ms])
static ColumnVector  input_torque (JOINT_NUM, 0.0);
//---------------------------------------------------------------------------------------------



void stepSimulation (const dReal &time_step)
{
  for (int j(0); j<JOINT_NUM; ++j)
    joint[j].addTorque(input_torque(j));
  // cerr<<"torque= "<<input_torque.transpose()<<endl;

  // シミュレーション
  space.collide (0,&nearCallback);
  world.step (time_step);
  // time += time_step;
  // remove all contact joints
  contactgroup.empty();
}
//---------------------------------------------------------------------------------------------


bool oct_robot_server (void)
{
  TXData data;
  while (1)
  {
    if (global_file_descriptor<0)
    {
      cerr<<"connection terminated (unexpected error)."<<data.command<<endl;
      exit(1);
    }
    read (global_file_descriptor, (char*)&data, sizeof(data));
    switch (data.command)
    {
      case  ORS_START_SIM         :
        return true;
      case  ORS_STOP_SIM          :
        return false;
      case  ORS_STEP_SIM          :
        stepSimulation (data.dvalue);
        break;
      case  ORS_RESET_SIM         :
        create_world();
        break;
      case  ORS_DRAW_WORLD        :
        return true;
      case  ORS_SET_TORQUE        :
        input_torque(data.step) = data.dvalue;
        break;
      case  ORS_SET_WINDOWSIZE    :
        if (data.step==0)  window_x=data.ivalue;
        else if (data.step==1)  window_y=data.ivalue;
        break;
      case  ORS_GET_JOINT_NUM     :
        write (global_file_descriptor, (char*)&JOINT_NUM, sizeof(JOINT_NUM));
        break;
      case  ORS_GET_JSTATE_DIM    :
        write (global_file_descriptor, (char*)&JOINT_STATE_DIM, sizeof(JOINT_STATE_DIM));
        break;
      case  ORS_GET_BSTATE_DIM    :
        write (global_file_descriptor, (char*)&BASE_STATE_DIM, sizeof(BASE_STATE_DIM));
        break;
      case  ORS_GET_JOINT_STATE  :
        getJointState (joint_state);
        // cerr<<"joint2= ";for(int j(0);j<JOINT_STATE_DIM;++j)cerr<<" "<<joint_state[j];cerr<<endl;
        write (global_file_descriptor, (char*)joint_state, sizeof(double)*JOINT_STATE_DIM);
        break;
      case  ORS_GET_BASE_STATE   :
        getBaseState (base_state);
        write (global_file_descriptor, (char*)base_state, sizeof(double)*BASE_STATE_DIM);
        break;
      default :
        cerr<<"in oct_robot_server(): invalid command "<<data.command<<endl;
        return false;
    }
  }
}
//-------------------------------------------------------------------------------------------

void setup_server (void)
  // ref. http://www.ueda.info.waseda.ac.jp/~toyama/network/example1.html
{
  int    fd1;
  struct sockaddr_un    saddr;
  struct sockaddr_un    caddr;
  int    len;

  if ((fd1 = socket (PF_UNIX, SOCK_STREAM, 0)) < 0)
  {
    perror("socket");
    exit(1);
  }

  bzero ((char *)&saddr, sizeof(saddr));
  // ソケットの名前を代入
  saddr.sun_family = AF_UNIX;
  strcpy (saddr.sun_path, SOCK_NAME);
  // ソケットにアドレスをバインド
  unlink(SOCK_NAME);
  if (bind(fd1, (struct sockaddr *)&saddr,
          sizeof(saddr.sun_family) + strlen(SOCK_NAME)) < 0){
    perror("bind");
    exit(1);
  }
  // listen をソケットに対して発行
  if (listen(fd1, 1) < 0)
  {
    perror("listen");
    exit(1);
  }

  len = sizeof(caddr);
  /*
    * accept()により、クライアントからの接続要求を受け付ける。
    * 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
    * fd2に返される。このfd2を通して通信が可能となる。
    * fd1は必要なくなるので、close()で閉じる。
    */
  if ((global_file_descriptor = accept(fd1, (struct sockaddr *)&caddr, (socklen_t*)&len)) < 0)
  {
      perror("accept");
      exit(1);
  }
  close(fd1);
}
//-------------------------------------------------------------------------------------------


//===========================================================================================
/*! ¥brief 描画(OpenGL)のコールバック関数.
    ¥param[in] pause 停止モードなら true (0以外)

    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,
    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.  */
static void simLoop (int pause)
//===========================================================================================
{
//   static dReal time(0.0); // シミュレーション時間
  if (!pause)
  {
    if (!oct_robot_server())  dsStop();
  }

  draw_world();
}
//-------------------------------------------------------------------------------------------

static void stopSimulation (void)
{
  close (global_file_descriptor);
  global_file_descriptor = -1;
}
//-------------------------------------------------------------------------------------------


int main (int argc, char **argv)
{
  dsFunctions fn; // OpenGL 出力用オブジェクト
  fn.version = DS_VERSION;
  fn.start = &start;
  fn.step = &simLoop;
  fn.command = &keyEvent;
  fn.stop = &stopSimulation;
  char path_to_textures[]="textures";
  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

  #if ODE_MINOR_VERSION>=9
  # if ODE_MINOR_VERSION==9
    dInitODE();
  # elif ODE_MINOR_VERSION>=10
    dInitODE2(0);
  # endif
  #endif

  setup_server();
  if (oct_robot_server())
  {
    create_world();
    // run simulation
    if(window_x>0 && window_y>0)
      dsSimulationLoop (argc,argv,window_x,window_y,&fn);
    else
      while(oct_robot_server());
  }
  if(global_file_descriptor!=-1)
  {
    close (global_file_descriptor);
    global_file_descriptor = -1;
  }

  #if ODE_MINOR_VERSION>=9
    dCloseODE();
  #endif
  return 0;
}
//-------------------------------------------------------------------------------------------

```

```
//-------------------------------------------------------------------------------------------
/*! ¥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);
```

```cpp
  static const double MaxTorque(100.0);  // [Nm]
  if(init)
  {
    const double q1=-0.25*M_PI, q2=0.5*M_PI;
    for(int i(0);i<8;i+=2) target(i)=q1;
    for(int i(1);i<8;i+=2) target(i)=q2;
    init = false;
  }
  ColumnVector u (JOINT_NUM,0.0);  // 制御入力（トルク）
  ColumnVector jstate (get_joint_state());  // 現在の関節状態
  for (int i(0);i<8;++i)
  {
    u(i)=kp*(target(i)-jstate(i))-kd*jstate(8+i);
    if (u(i)>MaxTorque)  u(i)=MaxTorque;
    else if (u(i)<-MaxTorque)  u(i)=-MaxTorque;
  }

  return u;
}
//--------------------------------------------------------------------------------------------

inline void start_simulation (int window_width, int window_height)
{
  chfd();
  TXData data;
  data.command = ORS_SET_WINDOWSIZE;
  data.step = 0;
  data.ivalue = window_width;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
  data.step = 1;
  data.ivalue = window_height;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
  data.command = ORS_START_SIM;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
}

inline void stop_simulation (void)
{
  chfd();
  TXData data;
  data.command = ORS_STOP_SIM;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
  close(client_file_descriptor);
  client_file_descriptor=-1;
}

inline void step_simulation (const ColumnVector &u, const double &time_step)
{
  chfd();
  TXData data;
  data.command = ORS_SET_TORQUE;
  for (int j(0); j<JOINT_NUM; ++j)
  {
    data.step = j;
    data.dvalue = u(j);
    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
  }
  data.command = ORS_STEP_SIM;
  data.dvalue = time_step;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
}

inline void reset_simulation (void)
{
  chfd();
  TXData data;
  data.command = ORS_RESET_SIM;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
}

inline ColumnVector get_joint_state (void)
{
  chfd();
  if (JOINT_STATE_DIM<=0)  return ColumnVector(0);
  if (jState.dim1() != JOINT_STATE_DIM)  jState.resize(JOINT_STATE_DIM);
  TXData data;
  data.command = ORS_GET_JOINT_STATE;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
  chret (read (client_file_descriptor, (char*)joint_state, sizeof(double)*JOINT_STATE_DIM));
  // cerr<<"c-joint1= ";for(int j(0);j<JOINT_STATE_DIM;++j)cerr<<" "<<joint_state[j];cerr<<endl;
  for (int j(0); j<JOINT_STATE_DIM; ++j)  jState(j)=joint_state[j];
  // cerr<<"c-joint2= "<<jState.transpose()<<endl;
  return jState;
}
inline ColumnVector get_base_state (void)
{
  chfd();
  if (BASE_STATE_DIM<=0)  return ColumnVector(0);
  if (bState.dim1() != BASE_STATE_DIM)  bState.resize(BASE_STATE_DIM);
  TXData data;
  data.command = ORS_GET_BASE_STATE;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
  chret (read (client_file_descriptor, (char*)base_state, sizeof(double)*BASE_STATE_DIM));
  for (int j(0); j<BASE_STATE_DIM; ++j)  bState(j)=base_state[j];
  return bState;
}

inline void draw_world (void)
{
  chfd();
  TXData data;
  data.command = ORS_DRAW_WORLD;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
}
```

```
inline int get_joint_num (void)
{
  chfd();
  TXData data;
  data.command = ORS_GET_JOINT_NUM;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
  chret (read (client_file_descriptor, (char*)&JOINT_NUM, sizeof(JOINT_NUM)));
  cerr<<"joint-num = "<<JOINT_NUM<<endl;
  return JOINT_NUM;
}

inline int get_joint_state_dim (void)
{
  chfd();
  TXData data;
  data.command = ORS_GET_JSTATE_DIM;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
  chret (read (client_file_descriptor, (char*)&JOINT_STATE_DIM, sizeof(JOINT_STATE_DIM)));
  cerr<<"joint-state-dim = "<<JOINT_STATE_DIM<<endl;
  if (joint_state!=NULL) {delete[] joint_state; joint_state=NULL;}
  joint_state = new double[JOINT_STATE_DIM];
  return JOINT_STATE_DIM;
}

inline int get_base_state_dim (void)
{
  chfd();
  TXData data;
  data.command = ORS_GET_BSTATE_DIM;
  chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
  chret (read (client_file_descriptor, (char*)&BASE_STATE_DIM, sizeof(BASE_STATE_DIM)));
  cerr<<"base-state-dim = "<<BASE_STATE_DIM<<endl;
  if (base_state!=NULL) {delete[] base_state; base_state=NULL;}
  base_state = new double[BASE_STATE_DIM];
  return BASE_STATE_DIM;
}

#ifdef OUTPUT_OCT
DEFUN_DLD (startSimulation, args, ,
           "int startSimulation(int window_width, int window_height).")
{
  if(!setup_client()) return octave_value(1);
  start_simulation(args(0).double_value(), args(1).double_value());
  get_joint_num();
  get_joint_state_dim();
  get_base_state_dim();
  return octave_value(0);
}

DEFUN_DLD (stopSimulation, args, ,
           "void stopSimulation(void).")
{
  stop_simulation();
  return octave_value();
}

DEFUN_DLD (stepSimulation, args, ,
           "void stepSimulation(const ColumnVector &u, const dReal &time_step).")
{
  ColumnVector u(args(0).vector_value());
  step_simulation (u, args(1).double_value());
  return octave_value();
}

DEFUN_DLD (resetSimulation, args, ,
           "void resetSimulation(void).")
{
  reset_simulation();
  return octave_value();
}

DEFUN_DLD (drawWorld, args, ,
           "void drawWorld(void).")
{
  draw_world();
  return octave_value();
}

DEFUN_DLD (getJointState, args, ,
           "ColumnVector getJointState(void).")
{
  ColumnVector state (get_joint_state());
  return octave_value (state);
}

DEFUN_DLD (getBaseState, args, ,
           "ColumnVector getBaseState(void).")
{
  ColumnVector state (get_base_state());
  return octave_value (state);
}
#endif


int main (int argc, char **argv)
{
  if(!setup_client()) return 1;
  start_simulation(400,400);
  get_joint_num();
  get_joint_state_dim();
  get_base_state_dim();
  while(1)
  {
    step_simulation (get_torque(), 0.001);
    draw_world();
  }

  close (client_file_descriptor);
  return 0;
}
//-------------------------------------------------------------------------------------------


```

```
function [phi] = getPhi(state,aind,centers,B,var,nactions)

    % 現在の状態に関する基底関数
    dist = sum((centers - repmat(state',B,1)).^2,2);

    phi = zeros(B*nactions,1);
    phi(B*(aind-1)+1:B*aind) = exp(-dist/2/var^2);
```

```
function theta=KernelLeastSquaresPolicyIteration(L, M, T, options, win_w,win_h)
  startSimulation (win_w,win_h);    % 本体のウィンドウを表示
  actions = [-50, 0, 50];           % 行動の候補
  nactions = 3;                     % 行動数

  % カネル行列K, ベクトルrの初期化
  K = zeros(M*T,M*T);
  r = zeros(M*T,1);

  % モデルパラメータの初期化
  theta = rand(M*T,1);

  % データ行列の初期化, 状態次元+行動次元=5
  data = zeros(M*T,5);

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

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

      for t=1:T
        % 状態（psi1, psi2, dpsi1, dpsi2）の観測
        state = getJointState();

        if l==1
          policy = ones(nactions,1)./nactions;
        else
          Q(1) = theta'*exp(-sum((pdata-repmat([state' actions(1)],M*T,1)).^2,2)/2/(options.var^2));
          Q(2) = theta'*exp(-sum((pdata-repmat([state' actions(2)],M*T,1)).^2,2)/2/(options.var^2));
          Q(3) = theta'*exp(-sum((pdata-repmat([state' actions(3)],M*T,1)).^2,2)/2/(options.var^2));

          % 政策
          policy = zeros(nactions);
          switch options.pmode
            case 1 % greedy
              [v,a] = max(Q);
              policy(a) = 1;

            case 2 % e-greedy
              [v,a] = max(Q);
              policy = ones(nactions,1)*options.epsilon/nactions;
              policy(a) = 1 - options.epsilon+options.epsilon/nactions;

            case 3  % softmax
              policy = exp(Q./options.tau)/sum(exp(Q./options.tau));
          end
        end

        % 行動選択
        ran = rand;
        if(ran < policy(1))
```

```
     action = 1;
    elseif(ran < policy(1)+policy(2))
     action = 2;
    else
     action = 3;
    end

    u(2) = actions(action);

    % 行動の実行
    stepSimulation (u,0.01);
    if(t==0 || mod(t,10)==0)
     drawWorld;
    end

    % データ行列の更新
    data(T*(m-1)+t,1) = state(1);
    data(T*(m-1)+t,2) = state(2);
    data(T*(m-1)+t,3) = state(3);
    data(T*(m-1)+t,4) = state(4);
    data(T*(m-1)+t,5) = u(2);

    % 状態（psi1，psi2，dpsi1，dpsi2）の観測
    state = getJointState();

    % M*T次元報酬ベクトルr
    r(T*(m-1)+t) = -cos(state(1));

    % 割引き和の計算
    dr = dr + options.gamma^(t-1) * r(T*(m-1)+t);
   end
  end

  % (M*T)*(M*T)カーネル行列の生成
  for mt=1:M*T-1
   K(mt,:) = exp(-sum((data(1:M*T,:)-repmat(data(mt,:),M*T,1)).^2,2)/2/(options.var^2))' - options↙
gamma * exp(-sum((data(1:M*T,:)-repmat(data(mt+1,:),M*T,1)).^2,2)/2/(options.var^2))';
  end

  % 最小二乗法による政策評価
  theta = pinv(K)*r;

  pdata = data;

  printf("%d)Max=%.2f Avg=%.2f Dsum=%.2f numtop=%d¥n",l,max(r),mean(r),dr/M,size(find(r>0.9),1));
  fflush(stdout);
 end
```

```
function [theta]=LeastSquaresPolicyIteration(L, M, T, B, options, win_w,win_h)
  startSimulation (win_w,win_h);    % 本体のウィンドウを表示
  actions = [-50, 0, 50];           % 行動の候補
  nactions = 3;             % 行動数

  % デザイン行列X，ベクトルrの初期化
  X = zeros(M*T,B*nactions);
  r = zeros(M*T,1);

  % モデルパラメータの初期化
  theta = zeros(B*nactions,1);

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

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

      for t=1:T+1

        % 状態（psi1，psi2，dpsi1，dpsi2）の観測
        state = getJointState();

        % 距離
        dist = sum((options.centers - repmat(state',B,1)).^2,2);

        % 現在の状態に関する基底関数
        phis = exp(-dist/2/(options.var^2));

        % 現在の状態における価値関数
        Q = phis'*reshape(theta,B,nactions);

        % 政策
        policy = zeros(nactions,1);
        switch options.pmode
          case 1 % greedy
            [v,a] = max(Q);
            policy(a) = 1;

          case 2 % e-greedy
            if l==1
              policy = ones(nactions,1)./nactions;
            else
              [v,a] = max(Q);
              policy = ones(nactions,1)*options.epsilon/nactions;
              policy(a) = 1 - options.epsilon+options.epsilon/nactions;
            end

          case 3  % softmax
            policy = exp(Q./options.tau)/sum(exp(Q./options.tau));
        end
        % 行動選択
        ran = rand;
        if(ran < policy(1))
          action = 1;
        elseif(ran < policy(1)+policy(2))
          action = 2;
        else
          action = 3;
        end
        u(2) = actions(action);

        % 行動の実行
        stepSimulation (u,0.005);
        if(t==0 || mod(t,10)==0)
           drawWorld;
        end

        if t>1
          % 現在の状態に関する基底関数の政策に関する平均
          aphi = zeros(B*nactions,1);
          for a=1:nactions
            aphi = aphi + getPhi(state,a,options.centers,B,options.var,nactions) * policy(a);
          end

          % 一つ前の状態と行動に関する基底関数
          pphi = getPhi(pstate,paction,options.centers,B,options.var,nactions);

          % (M*T)*Bデザイン行列X, M*T次元ベクトルr
          X(T*(m-1)+t-1,:) = (pphi - options.gamma * aphi)';
          r(T*(m-1)+t-1) = -cos(state(1));

          % 割引き和の計算
          dr = dr + r(T*(m-1)+t-1)*options.gamma^(t-1);
        end

        paction = action;
        pstate = state;
      end
    end

    % 政策評価
    theta = pinv(X'*X)*X'*r;

    printf("%d)Max=%.2f Avg=%.2f Dsum=%.2f numtop=%d¥n",l,max(r),mean(r),dr/M,size(find(r>0.9),1));
    fflush(stdout);
  end
```

```
//-------------------------------------------------------------------------------------------
/*! ¥file
    ¥brief  robot simulator protocol
    ¥author Akihiko Yamaguchi
    ¥date   Mar.13 2007  */
//-------------------------------------------------------------------------------------------

const char *SOCK_NAME = "/tmp/octrobot-socket";   //!< 通信に使うソケットファイル

// サーバ側が受け付けるコマンド
const int ORS_START_SIM          (0);  //!< シミュレーションを開始する
const int ORS_STOP_SIM           (1);  //!< シミュレーションを終了する
const int ORS_STEP_SIM           (2);  //!< シミュレーションを 1 ステップ進める （dvalue=時間ステップ）
const int ORS_RESET_SIM          (3);  //!< シミュレーションリセットする
const int ORS_DRAW_WORLD         (4);  //!< 世界を描画
const int ORS_SET_TORQUE         (5);  //!< トルク入力を設定 （step=0,..,関節数-1， dvalue）
const int ORS_SET_WINDOWSIZE     (6);  //!< 画面サイズを変更． デフォルト 400x400 （step=0:x， step=1:y， ivalue）．  ORS_START_SIM よりも前に使わないと意味がない．
const int ORS_GET_JOINT_NUM      (7);  //!< 関節数を返す． （int*1）
const int ORS_GET_JSTATE_DIM     (8);  //!< 関節状態の次元を返す． （int*1）
const int ORS_GET_BSTATE_DIM     (9);  //!< ベース状態の次元を返す． （int*1）
const int ORS_GET_JOINT_STATE   (10);  //!< 関節状態を取得 （double*関節状態ベクトル次元のデータが返される）
const int ORS_GET_BASE_STATE    (11);  //!< ベース状態を取得 （double*ベース状態ベクトル次元のデータが返される）

struct TXData  //! 通信に使うデータ
{
  int   command;
  int   step;
  union
  {
    int    ivalue;
    double dvalue;
  };
};
```

```
//-------------------------------------------------------------------------------------------
/*! ¥file
    ¥brief  create acrobot for ODE
    ¥author Akihiko Yamaguchi
    ¥date   Dec.26 2007 */
//-------------------------------------------------------------------------------------------
// dynamics and collision objects
static dWorld world;
static dSimpleSpace space (0);
static dPlane plane;
static dBody body[3];
static const int JOINT_NUM(2);
/*mod*/static const int JOINT_STATE_DIM(JOINT_NUM*2);  // 関節状態の次元
/*mod*/static const int BASE_STATE_DIM(13);  // ベース状態の次元
//*mod*/static const int STATE_DIM(4);
static dHingeJoint joint[JOINT_NUM];
static dFixedJoint base_joint; // 支柱を地面(z=0)に固定する
static dJointGroup contactgroup;
static dBox  LinkBase;
/*mod*/static dCapsule Link1, Link2;

const int  MAX_CONTACTS (10); // maximum number of contact points per body
//-------------------------------------------------------------------------------------------
/*mod*/static double joint_state[JOINT_STATE_DIM];
/*mod*/static double base_state[BASE_STATE_DIM];
//-------------------------------------------------------------------------------------------


//===========================================================================================
//! ¥brief シミュレーションオブジェクトを作成
void create_world( void )
//===========================================================================================
{
  // acrobot の回転軸（以下，支柱）は ここでは 直方体(Box)にしています.
  const dReal param_h0     = 0.05; // 支柱（直方体）の高さ[m]
  const dReal param_wx0    = 0.05; // 同幅(x)
  const dReal param_wy0    = 0.80; // 同幅(y)
  const dReal param_z0     = 1.20; // 支柱の垂直位置[m]

  const dReal param_l1     = 0.50; // 第1リンク（支柱に近いリンク）の長さ[m]
  const dReal param_d1     = 0.15; // 同直径[m]
  const dReal param_l2     = 0.50; // 第2リンク（支柱に近いリンク）の長さ[m]
  const dReal param_d2     = 0.15; // 同直径[m]

  const dReal density      = 1000.0;  // 各リンクの密度[kg/m^3]. 参考(?)`人体の密度' は 900~1100 kg/m^3 (wikipedia)

  int i;
  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）.

  i=0; {
    body[i].create (world);
    body[i].setPosition (0.0, 0.0, param_z0); // 支柱の中心座標
    dReal xx=param_wx0, yy=param_wy0, zz=param_h0;
    dMass m;
    m.setBox (density,xx,yy,zz);
    body[i].setMass (&m);
    LinkBase.create (space,xx,yy,zz);
    LinkBase.setBody (body[i]);
  }
  i=1; {
    body[i].create (world);
    body[i].setPosition (0.0, 0.0, param_z0-0.5*param_l1); // リンク1の中心座標
    dReal rad=0.5*param_d1, len=param_l1-2.0*rad;
    dMass m;
    m.setCappedCylinder (density,3,rad,len);  // direction(3): z-axis
    body[i].setMass (&m);
    Link1.create (space,rad,len);
    Link1.setBody (body[i]);
  }
  i=2; {
    body[i].create (world);
    body[i].setPosition (0.0, 0.0, param_z0-param_l1-0.5*param_l2); // リンク2の中心座標
    dReal rad=0.5*param_d2, len=param_l2-2.0*rad;
    dMass m;
    m.setCappedCylinder (density,3,rad,len);  // direction(3): z-axis
    body[i].setMass (&m);
    Link2.create (space,rad,len);
    Link2.setBody (body[i]);
  }

  i=0; {
    const dReal *pos = body[0].getPosition();
    joint[i].create (world);
    joint[i].attach (body[0],body[1]);
    joint[i].setAnchor (pos[0],pos[1],pos[2]); // 回転中心=支柱の中心(=原点)
    joint[i].setAxis (0.0,1.0,0.0); // 回転軸=y軸
    // joint[i].setParam (dParamHiStop, +0.5*M_PI); // 関節の可動範囲を制約するときに使う
    // joint[i].setParam (dParamLoStop, -0.5*M_PI); // acrobot の場合は省略
  }
  i=1; {
    const dReal *pos = body[1].getPosition();
    joint[i].create (world);
    joint[i].attach (body[1],body[2]);
    joint[i].setAnchor (pos[0],pos[1],pos[2]-0.5*param_l1); // 回転中心=リンク1とリンク2の間
    joint[i].setAxis (0.0,1.0,0.0); // 回転軸=y軸
  }
  base_joint.create(world);
  base_joint.attach(body[0].id(),0); // 支柱(body[0]) と 平面(0)の間の固定リンク. 支柱が固定される.
  base_joint.set();
}
//-------------------------------------------------------------------------------------------

```

```

//============================================================================================
//! ¥brief 描画関数
void draw_world( void )
//============================================================================================
{
  int i;
  dsSetColor (0,0.5,1);
  dsSetTexture (DS_WOOD);
  dReal rad, len;
  dReal sides[4];
  dBox *blink;
  dCapsule *clink;
  dsSetColorAlpha (1.0, 0.0, 0.0, 0.6);
  i=0; blink=&LinkBase; blink->getLengths(sides); dsDrawBox (body[i].getPosition(),body[i].getRotation(),sides);
  dsSetColorAlpha (0.0, 0.5, 1.0, 0.6);
  i=1; clink=&Link1; clink->getParams(&rad, &len); dsDrawCappedCylinder (body[i].getPosition(),body[i].getRotation(),len,rad);
  i=2; clink=&Link2; clink->getParams(&rad, &len); dsDrawCappedCylinder (body[i].getPosition(),body[i].getRotation(),len,rad);
}
//--------------------------------------------------------------------------------------------




```

```
function test()
  if(startSimulation(400,400)~=0) return; endif % シミュレーション開始(ウィンドウサイズ)
  more off
  printf('press any key..¥n')
  kbhit();
  tidx = 9;
  printf('press q: quit¥n')
  K = [500,0,50,0; 0,500,0,50; 0,0,0,0; 0,0,0,0];
  target = [
    0,      0,  0,  0;
    0.46, -1.83,  0,  0;
    0.06, -1.07,  0,  0;
    0.12,   2.01,  0,  0;
    0.67, -0.08,  0,  0;
    0.65, -2.53,  0,  0;
   -1.76, -3.10,  0,  0;
   -0.02, -3.04,  0,  0;
    1.71, -2.55,  0,  0;
   -0.01,   1.01,  0,  0;
   -1.28,   3.00,  0,  0;
   -2.86,   2.00,  0,  0;
   -2.30,   1.10,  0,  0;
   -1.00, -0.90,  0,  0;
    0.02,   0.95,  0,  0;
    1.52,   0.1,  0,  0;
    3.13, -2.6,  0,  0;
  ];
  MaxTorque = 200.0;
  while (1)
    state = getJointState();% acrobot の [q0,q1,dq0,dq1]' を返す
    u = K * (target(tidx,:)' - state);
    stepSimulation (u,0.005); % トルク, 時間幅
    key=kbhit(1);
    switch key
      case 'q';  stopSimulation(); printf('¥n'); return;    % シミュレーションを終了
      case 'n';  tidx = tidx + 1; printf('target: %.2f %.2f¥n',target(tidx,1),target(tidx,2));
    endswitch
    drawWorld();
  endwhile
```