```
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
```