```
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
```