AOloopRW function

```
function [var_eps] = AOloopRW(G,H, covariance_phi, sigma_e, phi_sim)
% Variance calculation of an AO system in the closed-loop
 configuration
% with the Random Walk model used for calculations
% IN
% G      : measurement matrix
% H      : influence matrix mapping the wavefront on the mirror
% covariance_phi  : covariance matrix of the turbulence wavefront
% sigma_e : measurement noise parameter for determining its covariance
% phi_sim : simulation data for the wavefront
% OUT
% var_eps : variance of the residual wavefront after taking N_t points
% within the closed-loop operation

% dimension lifted wavefront
n_H = size(H,1);

% dimension lifted sensor slopes
n_G = size(G,1);

% Number of sample points for phi_sim
T = length(phi_sim);

u = zeros(n_H,T);

% This term is multiplied with the slope vector to obtain the
 eps_hat(k|k)
% value
eps_pred_multiplier_matrix = (H'*H)\H'*(covariance_phi*G'/
(G*covariance_phi*G' + (sigma_e^2)*eye(n_G)));

% epsilon matrix
eps_k = zeros(size(phi_sim,1),T);

% epsilon matrix with mean removed:
eps_mean_removed_k = zeros(size(phi_sim,1),T);

% Constructing a vector of all the variance values for eps
var_eps = zeros(T,1);

% An assumption is that H is full rank.

% We have the data for phi_k, and we know the matrix H.
% Since we don't have the values for the slopes s(k), we compute the
 vector as:
% s(k) = G*phi_k - G*H*u(k-1) + sigma_e*randn(n_G,1)
% We use this s(k) in the expression for the optimal increment u(k) -
% u(k-1), and calculate the values of the u(k) vector recursively,
 since we
% know u(0) = 0.
```

```
% Based on the calculated u(k), we calculate the value of
% epsilon and then accordingly its variance.

% u(0) is 0 since we don't apply any control input before the first
 wavefront datum,
% and hence we calculate u(1) taking u(0) = 0
u(:,1) = eps_pred_multiplier_matrix*(G*phi_sim(:,1) +
 sigma_e*randn(n_G,1));
eps_k(:,1) = phi_sim(:,1);

eps_mean_removed_k(:,1) = eps_k(:,1) - mean(eps_k(:,1));

var_eps(1) = var(eps_mean_removed_k(:,1));

for k = 2:T
    eps_k(:,k) = phi_sim(:,k) - H*u(:,k-1);
    u(:,k) = eps_pred_multiplier_matrix*(G*eps_k(:,k) +
 sigma_e*randn(n_G,1)) + u(:,k-1);
    eps_mean_removed_k(:,k) = eps_k(:,k) - mean(eps_k(:,k));
    var_eps(k) = var(eps_mean_removed_k(:,k));
end

var_eps = mean(var_eps);

end
```

Published with MATLAB® R2018a