AOloopSID function

```
function [var_eps, VAF] = AOloopSID(G,H,As,Cs,Ks,sigma_e,phi_sim)
% Variance calculation of an AO system in the closed-loop
 configuration,
% using the Subspace Identification model
% IN
% G     : measurement matrix
% H     : influence matrix mapping the wavefront on the mirror
% As    : State matrix for turbulence model
% Cs    : State matrix for turbulence model
% Ks    : Kalman Gain for turbulence model
% sigma_e : measurement noise parameter
% 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
% VAF: Variance accounted for by the turbulence model

% State Dimension
n = size(As,1);

% 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);

% Control Input u
u = zeros(n_H,T);

% 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);

% VAF
VAF = zeros(T,1);

% An assumption is that H is full rank.

% We use s(k) in the equation to find optimal prediction x(k+1|k).
% the optimal prediction for s(k+1|k) is hence found as Cs*x(k+1|k)
% We can now find prediction for phi_k using the expression derived in
 Part
```

```
% 1.1.
% Hence, a minimizing u(k) can be found by solving a least squares
 problem to
% minimize the predicted residual (epsilon(k+1|k)).
% This u(k) will be used to calculate the actual residual epsilon(k)
 and
% then accordingly its variance.

% u(0) is 0 since we don't apply any control input before the first
% wavefront datum.

eps_k(:,1) = phi_sim(:,1);

% We have the data for phi_sim
% We compute the measurements for s as:
s_sim = G*phi_sim + sigma_e*randn(n_G,T);
%s_current = G*eps_k(:,1) + sigma_e*randn(n_G,1);

% (Taking initial x as zero)
x_current = zeros(n,1);
x_pred = (As-Ks*Cs)*x_current + Ks*(s_sim(:,1));
s_pred = Cs*x_pred;
% We use the same method as 1.1 to get a pseudo inverse of G. This is
% equivalent to pinv()
Ginv = pinv(G);
phi_pred = Ginv*s_pred;

% Finally, calculating input u:
Hterm = inv((H')*H)*(H');
u(:,1) = Hterm*phi_pred;

% Mean removal of epsilon
eps_mean_removed_k(:,1) = eps_k(:,1) - mean(eps_k(:,1));
var_eps(1) = var(eps_mean_removed_k(:,1));

actual_phi_norm_sum = 0; % For VAF calculation
deviation_norm_sum = 0; % For VAF calculation

for k = 2:T-1
    % Calculate actual residual epsilon
    eps_k(:,k) = phi_sim(:,k) - H*u(:,k-1);
    %s_current = G*eps_k(:,k) + sigma_e*randn(n_G,1);

    % Calculate next u based on predicted epsilon
    x_current = x_pred;
    x_pred = (As-Ks*Cs)*x_current + Ks*(s_sim(:,k));
    s_pred = Cs*x_pred;
    phi_pred = Ginv*s_pred;

    u(:,k) = Hterm*phi_pred;

    % Mean removal and Variance
    eps_mean_removed_k(:,k) = eps_k(:,k) - mean(eps_k(:,k));
    var_eps(k) = var(eps_mean_removed_k(:,k));
```

```
    % VAF calculation
    phi_pred_mean_removed = phi_pred - mean(phi_pred);
    phi_sim_mean_removed = phi_sim(:,k+1) - mean(phi_sim(:,k+1));
    current_norm = (norm(phi_sim_mean_removed -
 phi_pred_mean_removed))^2;
    deviation_norm_sum = deviation_norm_sum + current_norm;
    actual_phi_norm_sum = actual_phi_norm_sum +
 (norm(phi_sim_mean_removed))^2;
end

var_eps = mean(var_eps);

mean_deviation_norm = deviation_norm_sum/(T-2);
mean_actual_norm = actual_phi_norm_sum/(T-2);
VAF = max(0,100*(1 - mean_deviation_norm/mean_actual_norm));

end
```

Published with MATLAB® R2018a