Table of Contents


```
% Filtering and Identification - final assignment
% Section 4 - Critical thinking

close all;

load systemMatrices.mat;
load turbulenceData.mat; % load the data provided

% Considering the first dataset for this section
phi_sim = phiSim{1,1};

phi_size = size(phi_sim,1);
```

Question 4.1

```
% Rank of G*H
rank_GH = rank(G*H);

% G*H is not full-rank; rank = 47
[U, S, V] = svd(G*H);
U1 = U(:,1:47);
V1 = V(:,1:47);
Sigma_SVD = S(1:47,1:47);

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

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

covariance_phi = zeros(phi_size, phi_size);

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

u = zeros(n_H,T);
for k = 1:T
    covariance_phi = covariance_phi + (phi_sim(:,k)*phi_sim(:,k)');
end

covariance_phi = covariance_phi/T;

s_k = zeros(n_G,length(phi_sim));
```

```
eps_k = zeros(n_H,length(phi_sim));

s_k(:,1) = G*phi_sim(:,1) + sigmae*randn(n_G,1);

eps_pred_multiplier_matrix = (covariance_phi*G'/(G*covariance_phi*G' +
 (sigmae^2)*eye(n_G)));

% Linear least-squares solution
u(:,1) = (V1/Sigma_SVD)*U1'*(G*eps_pred_multiplier_matrix*s_k(:,1));

var_s = zeros(T,1);

for k = 2:T
    s_k(:,k) = G*(phi_sim(:,k) - H*u(:,k-1)) + sigmae*randn(n_G,1);
    Y_k = G*(eps_pred_multiplier_matrix*s_k(:,k) + H*u(:,k-1));
    u(:,k) = (V1/Sigma_SVD)*U1'*Y_k;
    eps_k(:,k) = phi_sim(:,k) - H*u(:,k-1);

    var_s(k) = var(eps_k(:,k) - mean(eps_k(:,k)));

end

var_s = mean(var_s);
```

Question 4.2

```
% VAF calculation

deviation_norm_sum = 0;
actual_phi_norm_sum = 0;

for k = 2:T-1
    % predicted_phi = eps_predicted + Hu(k)
    predicted_phi = eps_pred_multiplier_matrix*s_k(:,k) + H*u(:,k-1);
    predicted_phi = predicted_phi - mean(predicted_phi);
    phi_sim_mean_removed = phi_sim(:,k+1) - mean(phi_sim(:,k+1));
    current_norm = (norm(phi_sim_mean_removed - predicted_phi))^2;
    deviation_norm_sum = deviation_norm_sum + current_norm;
    actual_phi_norm_sum = actual_phi_norm_sum +
 (norm(phi_sim_mean_removed))^2;
end

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));
```

Question 4.3

```
phi_vec = null(G);
%phi_vec = phi_sim(:,1);
phi_matrix_1 = zeros(7,7);
phi_matrix_2 = zeros(7,7);
for i = 1:7
```

```
    phi_matrix_1(:,i) = phi_vec(7*(i-1) + 1:7*i,1);
    phi_matrix_2(:,i) = phi_vec(7*(i-1) + 1:7*i,2);
end

% To see how the turbulent wavefront needs to be to lie in the
 nullspace of
% G
figure(1);
imagesc(phi_matrix_1);

figure(2);
imagesc(phi_matrix_2);
```

Published with MATLAB® R2018a