SC42025 FILTERING AND IDENTIFICATION

FINAL ASSIGNMENT

TURBULENCE MODELING FOR ADAPTIVE OPTICS

Aniket Ashwin Samant, Snehal Jauhri

(Student IDs: 4838866, 4772202)

January 18, 2019

Contents

INTRODUCTION

This assignment deals with modeling an *Adaptive Optics* (AO) system in which three different data-driven turbulence modeling methods are used to achieve optimal control performances, viz.

- a random-walk process
- a Vector-Auto-Regressive model
- a stochastic state-space model

Each model has some questions associated with it, and we solve them in chronological sequence taking one model at a time.

1. RANDOM WALK MODEL

We know from the assignment's equation (2) that:

$$s_o(k) = G\phi(k) + e(k)$$

We have the values of the wavefront sensor data in open-loop, $s_o(k)$, and also the value of the matrix G. To compute the value of $\hat{\phi}(k)$, given no prior information on it, we follow the linear least-squares approach:

First, we determine whether the matrix G is full-rank or not. We load the *systemMatrices.mat* file which contains the matrix G, and then run the *rank* command in MATLAB to get a rank value of **47**, which is less than *min(number of rows, number of columns)* of G. Thus, we need to employ a linear least-squares method that doesn't assume the matrix G to be full-rank.

We know that there are multiple solutions to this problem, and for uniqueness we go with one such that the optimal solution, $\hat{\phi}(k)$, has a minimal 2-norm, thus leading to the original linear least-squares problem being reformulated as:

$$\min_{\phi(k)\in\Gamma} \|\phi(k)\|_2^2 \text{ with } \Gamma = \left\{\phi(k) : \phi(k) = arg\min_{z} \|Gz - s_o(k)\|_2^2\right\}$$

By performing an SVD operation on the matrix G, we obtain:

$$G = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = U_1 \Sigma V_1^T$$

Here, $\Sigma \in \Re^{47x47}$ is non-singular, by the definition of SVD.

Now, let us define a partitioned vector,

$$\begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} = \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} z$$

Thus, our problem becomes:
$\underset{\xi_1}{min} \|U_1 \Sigma \xi_1 z - s_o(k)\|_2^2$
We get $\hat{\xi}_1 = \Sigma^{-1} U_1^T s_o(k)$, since Σ is a non-singular matrix. ξ_2 has no effect on the above expression and can be chosen arbitrarily. Thus, we get the optimal solution,

$$\hat{z} = \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} \hat{\xi}_1 \\ \hat{\xi}_2 \end{bmatrix} = V_1 \Sigma^{-1} U_1^T s_o(k) + V_2 \hat{\xi}_2$$

Since we're choosing a vector $\phi(k)$ with the smallest 2-norm, and $V_1^T V_2 = 0$ we get:
$\|\phi(k)\|_2^2 = \|V_1 \Sigma^{-1} U_1^T s_o(k)\|_2^2 + \|V_2 \hat{\xi}_2\|_2^2$
As we're minimizing the norm, we take $\hat{\xi}_2 = 0$ and we finally get the value of $\hat{\phi}(k)$ to be:

$$\hat{\phi}(\mathbf{k}) = \mathbf{V_1 \Sigma^{-1} U_1^T s_o}(\mathbf{k})$$

Question 2

We are provided with some prior information about the wavefront, viz.:

- E[ϕ(k)] = 0
- E[$\phi(k)\phi(k)^T$] = $C_\phi(0)$
- noise variance = σ_e^2

Based on equation (8) from the assignment, we approximate the value of $C_\phi(0)$ as:

$$C_\phi(0) = \frac{1}{N} \sum_{i=1}^{N} \phi(i)\phi(i)^T \tag{1}$$

We have the data necessary to formulate our problem of determining $\phi(k)$ as a stochastic linear least-squares problem, and hence we define our linear estimator $\tilde{\phi}(k)$ accordingly:

$$\tilde{\phi}(k) = \begin{bmatrix} M & N \end{bmatrix} \begin{bmatrix} s_o(k) \\ E[\phi(k)] \end{bmatrix}$$

such that $\mathrm{E}\left[(\tilde{\phi}(k)-\phi(k))(\tilde{\phi}(k)-\phi(k))^T\right]$ is minimized and $\mathrm{E}[\tilde{\phi}(k)] = \mathrm{E}[\phi(k)] = 0$

Thus, from the assignment's equation (2), based on $s_o(k)$'s expression, we can say,

$$\tilde{\phi}(k) = MG\phi(k) + Me(k) + NE\left[\phi(k)\right]$$

Since $\mathrm{E}[\tilde{\phi}(k)] = \mathrm{E}[\phi(k)] = 0$, we have:

$$\tilde{\phi}(k) = MG\phi(k) + Me(k)$$

Furthermore,

$$\phi(k) - \tilde{\phi}(k) = (I - MG)\phi(k) - Me(k)$$

Thus, the covariance of the above expression is computed as:

$$E\left[\left(\phi(k)-\tilde{\phi}(k)\right)\left(\phi(k)-\tilde{\phi}(k)\right)^T\right] = E\left[\left((I-MG)\phi(k)-Me(k)\right)\left((I-MG)\phi(k)-Me(k)\right)^T\right]$$

On expanding the right side of the equation and taking the error e(k) to be uncorrelated with the wavefront vector $\phi(k)$, and based on the statistical data provided to us, we get the following expression:

$$E\left[\left(\phi(k)-\tilde{\phi}(k)\right)\left(\phi(k)-\tilde{\phi}(k)\right)^T\right] = (I-MG)C_\phi(0)(I-MG)^T + M\sigma_e^2 I M^T$$

On further factorization, using the Schur complement of the factorized version, and the application of the "completion of squares" argument to the resulting equation, we get the optimum value of the matrix M that minimizes the covariance expression as:

$$M = C_\phi(0)G^T(GC_\phi(0)G^T + \sigma^2 I)^{-1}$$

(A point to note here: the term in brackets is invertible since $C_\phi(0)$ is a positive definite matrix and $\sigma^2 I$ is non-singular)

Accordingly, we also get the optimum estimate of the wavefront vector,

$$\hat{\phi}(k|k) = C_\phi(0)G^T(GC_\phi(0)G^T + \sigma_e^2 I)^{-1}s_o(k)$$

For questions 3 to 5, we assume $E[\epsilon(k)] = 0$ and $E[\epsilon(k)\epsilon(k)^T] = C_\phi(0)$

Question 3

Now, we consider the closed-loop system, and proceed to derive a UMVE of $\epsilon(k)$ using the given measurement set s(k). As in the previous question, we are provided with some prior information about the wavefront, viz.:

- $\mathrm{E}[\epsilon(k)] = 0$
- $\mathrm{E}\left[\epsilon(k)\epsilon(k)^T\right] = C_\phi(0)$
- noise variance = σ_e^2

We can clearly see that the equations (2) and (5) in the given assignment are similar, and we are told that the statistical data (the wavefront's and noise's mean and covariance values) are the same. Hence, as in the previous question, the optimum estimate of the wavefront vector is quite similar in the closed-loop system, and the only difference is in the value of the output vector, which in this case will be the closed-loop slope vector s(k):

$$\hat{\epsilon}(k|k) = C_\phi(0)G^T(GC_\phi(0)G^T + \sigma_e^2 I)^{-1}s(k)$$

Question 4

In this question, we make use of the random walk model represented by equation (9) in the assignment.

We know that:

$$\begin{aligned}\epsilon(k) &= \phi(k) - Hu(k-1)\\ \implies \phi(k) &= \epsilon(k) + Hu(k-1)\\ \implies \phi(k+1) &= \epsilon(k+1) + Hu(k)\end{aligned}$$

From the random walk model, we can relate $\phi(k)$ and $\phi(k+1)$, and substituting the above expressions respectively yields:

$$\epsilon(k+1) + Hu(k) = \epsilon(k) + Hu(k-1) + \eta(k)$$

If we consider $\hat{\epsilon}(k|k)$ to be the current optimal estimate, we know that the optimal one-step ahead prediction should be such that $\eta(k)$ is minimized, and based on which the input vectors must be chosen. That is to say,

$$\hat{\epsilon}(k+1|k) = \hat{\epsilon}(k|k) + Hu(k-1) - Hu(k) \tag{2}$$

Question 5

We denote $\delta u(k) := u(k) - u(k-1)$. We know from the previous question's equation 2 that:

$$\hat{\epsilon}(k+1|k) = \hat{\epsilon}(k|k) - H\delta u(k)$$

Thus, the minimization problem as described in the assignment's equation (6) can be reformulated as:

$$\underset{\delta u(k)}{min} \|\hat{\epsilon}(k|k) - H\delta u(k)\|_2^2$$

This is clearly a linear least-squares problem, and we know from running the *rank* command on the matrix H that it is full-rank. Hence, we can say that the optimal increment, $\hat{\delta} u(k)$ for minimizing the 2-norm is:

$$\hat{\delta} u(k) = (H^T H)^{-1} H^T \hat{\epsilon}(k|k)$$

We computed the expression for $\hat{\epsilon}(k|k)$ in question 3, and substituting the expression in the above equation yields the value of $\hat{\delta} u(k)$ as:

$$\hat{\delta} u(k) = (H^T H)^{-1} H^T C_\phi(0) G^T (G C_\phi(0) G^T + \sigma_e^2 I)^{-1} s(k) \tag{3}$$

Question 6

Given all the data, we compute the values for u(k) recursively based on the values of u(k-1) as defined by the relation in equation 3. We take u(1) to be 0, since we assume that for the first iteration, there's no input actuation applied through the deformable mirror. Accordingly, we get:

$$\epsilon(1) = \phi_{sim}(1)$$

Moreover, since we're provided with ϕ_{sim} values but not the values for the actual slope measurements s(k), we compute s(k) as:

$$s(k) = G\epsilon(k) + \sigma_e * randn(N_s, 1)$$

where `randn()` is a MATLAB function used for generating a normally distributed random noise vector. This s(k) is used in the computation of the optimum u(k) value. We iterate over

all the time samples to get the u(k) matrix for all input vectors, and once we have these values, we can calculate the ϵ matrix for all time samples based on:

$$\epsilon(k) = \phi(k) - Hu(k-1)$$

We apply the correction process of subtracting the mean vector from $\epsilon(k)$ and then calculate the variance of the vector. The variance for each time sample is calculated accordingly and the average variance of all time samples is returned by the function `AOloopRW()`.

Question 7

We compare the variance values returned by the random-walk model-based control design and the open-loop case (in which no control is applied). The variance returned by the former is around a value of 6.5 whereas the latter's returned value is around 22. We can clearly see a reduction in the variance of $\epsilon(k)$ values after controlling the residual wavefront using the DM.

The VAF is calculated based on another function, `VAF_RW` in which the $\phi(k+1|k)$ values are compared against $\phi_{sim}(k+1)$ values (after taking into consideration the mean value subtraction as performed in Question 6). The value returned is around 70%.

2. VAR Model of Order 1

We are provided with a VAR model represented by:

$$\phi(k+1) = A\phi(k) + w(k) \tag{4}$$

We have the following information provided to us to demonstrate that w(k) is uncorrelated with the measurement noise and the wavefront:

- $E\left[w(k)e(k)^T\right] = 0$
- $E\left[w(k)\phi(k)^T\right] = 0$

We are also provided with statistical information about w(k): w(k) $\sim \mathcal{N}(0, C_w)$

Question 1

We have the following data:

- $C_\phi(0) = E\left[\phi(k)\phi(k)^T\right]$
- $C_\phi(1) = E\left[\phi(k+1)\phi(k)^T\right]$

Multiplying equation 4 with $\phi(k)^T$ on both sides yields:

$$\phi(k+1)\phi(k)^T = A\phi(k)\phi(k)^T + w(k)\phi(k)^T$$

Further, on taking the expectation:

$$E\left[\phi(k+1)\phi(k)^T\right] = E\left[A\phi(k)\phi(k)^T\right] + E\left[w(k)\phi(k)^T\right]$$

Thus, we get the relation:

$$C_\phi(1) = AC_\phi(0)$$

and hence we calculate:

$$A = C_\phi(1)C_\phi(0)^{-1}$$

Question 2

We make an assumption here that $\phi(k)$ is a WSS signal.

Multiplying equation 4 with $w(k)^T$ on both sides, and then taking the resulting expectation yields:

$$E\left[\phi(k+1)\,w(k)^T\right] = E\left[A\phi(k)\,w(k)^T\right] + E\left[w(k)\,w(k)^T\right]$$

Based on the data we have, we get the following relation based on the above equation:

$$E\left[\phi(k+1)\,w(k)^T\right] = C_w$$

Taking the transpose of equation 4, and multiplying with $\phi(k+1)$ on both sides, we get:

$$E\left[\phi(k+1)\phi(k+1)^T\right] = E\left[A\phi(k+1)\phi(k)^T\right] + E\left[\phi(k+1)\,w(k)^T\right]$$

Since we have assumed $\phi(k)$ to be WSS, we can say that:
$E\left[\phi(k+1)\phi(k+1)^T\right] = E\left[\phi(k)\phi(k)^T\right] = C_\phi(0)$

And we know from the previous question that $C_\phi(1) = AC_\phi(0)$
Thus,

$$\begin{aligned} C_\phi(0) &= AC_\phi(1) + C_w \\ \implies C_\phi(0) &= A^2C_\phi(0) + C_w \end{aligned}$$

So we derive the following relationship:

$$C_w = (I - A^2)C_\phi(0)$$

Question 3

We need to formulate a state-space model with $\epsilon(k)$ being the state vector and s(k) being the output.

Based on equation (4) from the assignment, we can write:

$$\begin{aligned} \phi(k) &= \epsilon(k) + Hu(k-1) \\ \implies \phi(k+1) &= \epsilon(k+1) + Hu(k) \end{aligned}$$

Substituting this expression for $\phi(k)$ in equation (10), we get:

$$\epsilon(k+1) + Hu(k) = A\epsilon(k) + AHu(k-1) + w(k)$$
$$\implies \epsilon(k+1) = A\epsilon(k) + AHu(k-1) - Hu(k) + w(k)$$

Combining the above relation with equation (5) from the assignment, we have the following state-space model as required:

$$\epsilon(k+1) = A\epsilon(k) + \xi(k) + w(k)$$
$$s(k) = G\epsilon(k) + e(k)$$
$$where$$
$$\xi(k) = AHu(k-1) - Hu(k)$$

Question 4

Based on the state-space formulation from Question 3, we can express the Kalman filter in the observer form as:

$$\hat{\epsilon}(k+1) = (A - KG)\epsilon(k) + AHu(k-1) - Hu(k) + Ks(k)$$

We make use of the following DARE to compute the covariance matrix of $\epsilon(k)$, P, to which the optimal estimate of $\hat{P}(k+1|k)$ converges on solving the Kalman filter problem, as k $\rightarrow \infty$:

$$P = APA^T + Q - (APG^T)(GPG^T + R)^{-1}(APG^T)^T$$
$$where:$$
$$Q = C_w$$
$$R = E[v(k)v(k)^T]$$

An important point to note here is that v(k) and w(k) are taken to be uncorrelated, and the matrices A and G to be time-invariant.

Based on the P matrix calculated on solving the above Riccati equation, we compute the value of the stationary Kalman gain,

$$K = APG^T(GPG^T + R)^{-1}$$

Question 5

The minimum unbiased variance estimate for a state $x(k+1)$ is given by the following expression, which is derived based on the conditions of the conventional Kalman filtering problem

(here, we assume a stationary Kalman filter):

$$\hat{x}(k+1|k) = Ky(k) + Bu(k) + (A - KC)\hat{x}(k|k-1)$$

Here, x(k) is the state vector, y(k) is the output, and A,B, and C are the state-space matrices. In our case, based on our LTI state-space model with $\epsilon(k)$ being the state vector, s(k) being the output vector, and A and G being the state-space matrices, we compute the expression for the optimal one step ahead prediction recursively as:

$$\hat{\epsilon}(k+1|k) = Ks(k) + AHu(k-1) - Hu(k) + (A - KG)\hat{\epsilon}(k|k-1) \tag{5}$$

Question 6

Based on the control law represented by equation (6) in the assignment, we need to minimize the value of $\|\hat{\epsilon}(k+1|k)\|_2^2$ by controlling the input values u(k). From equation 5, we know that we need to minimize the value of:

$$\|(Ks(k) + (A - KG)\hat{\epsilon}(k|k-1)) + AHu(k-1) - Hu(k)\|_2^2 \tag{6}$$

Let us take $\xi(k) = Hu(k) - AHu(k-1)$
In the above equation, let us denote:
$Y(k) = Ks(k) + (A - KG)\hat{\epsilon}(k|k-1) + AHu(k-1)$
Hence, our minimization problem becomes:

$$\min_{u(k)} \|Y(k) - Hu(k)\|_2^2 \tag{7}$$

Knowing H to be full-rank and square, we have the linear least-squares solution,
$\hat{u}(k) = H^{-1}Y(k)$
That is to say,

$$\hat{u}(k) = H^{-1}(Ks(k) + (A - KG)\hat{\epsilon}(k|k-1)) + H^{-1}AHu(k-1) \tag{8}$$

Hence the optimum input vectors can be calculated recursively.

Question 7

The first function, `computeKalmanAR()` is used for calculating the values of A, C_w, and K using expressions derived previously in questions 1,2, and 4 respectively.

The function `AOloopAR()` is used for calculating the values of u(k) for which the 2-norm of $\epsilon(k+1|k)$ is minimum. It is performed using the expression derived in Question 6, and taking $\hat{\epsilon}(1|0)$ and u(0) to be 0.

u(k) and $\hat{\epsilon}(k+1|k)$ for the remaining indices can be calculated based on the recursive expressions for the two quantities. Once we have the u(k) values, the $\epsilon(k)$ values can be calculated as $\phi_{sim}(k) - Hu(k-1)$, and accordingly the average variance value can be calculated, as was done in Question (1.) 6 of the random walk model approach.

The variance value of ϵ returned by `AOloopAR()` is around 3.5.

Question 8

We compare the average variance value obtained in this case as against in the random walk model's. The value is further reduced using this model, and we see that this model outperforms the random walk model in terms of minimizing the variance of the estimates.

We compute the VAF for the turbulence wavefront one step ahead predicted values as in the previous case by comparing the predicted $\hat{\phi}(k+1|k)$ vectors against the $\phi_{sim}(k+1)$ vectors, and obtain a value of around 83 %.

3. SUBSPACE IDENTIFICATION

We are provided with a turbulence state space model description with a general state vector x and are tasked with finding the Matrices A_s, C_s and the Kalman gain K_s of the system in innovation form.

$$\begin{aligned} x(k+1) &= A_s x(k) + K_s v(k) \\ s(k) &= C_s x(k) + v(k) \end{aligned} \tag{9}$$

We generate the identification data i.e. $s_i d$ using the provided ϕ_{id} data.

Question 1

We use the N4SID method to estimate the State matrices and the state sequence. This is to ensure that we get the matrices and the Kalman gain up to the same similarity transformation.
Data Equation:

$$Y_{s,s,N} = O_s X_{s,N} + \tau_{v,s} V_{s,s,N} \tag{10}$$

Where Y, X and V are the Hankel Matrices and O_s is the observability matrix. τ is the Toeplitz matrix for the innovation signal.
Instrumental Variable:
The instrumental variable for our data equation is chosen as:

$$Z_N = Y_{0,s,N}$$

This Z_N thus eliminates $V_{s,s,N}$ from the data equation and gives:

$$\underset{N\to\infty}{Lt} \frac{1}{N} Y_{s,s,N} Y_{o,s,N}^T = \underset{N\to\infty}{Lt} \frac{1}{N} O_s X_{s,N} Y_{o,s,N}^T$$

We use an RQ factorization of the Instrumental variable for efficiency:

$$\begin{bmatrix} Y_{0,s,N} \\ Y_{s,s,N} \end{bmatrix} = \begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}$$

And so,

$$O_s X_{s,N} \approx R_{21} R_{11}^{-1} Z_N \tag{11}$$

The rank of the above expression (11) is n.
Hence we can use this RQ approximation and take it's SVD:

$$R_{21}R_{11}^{-1}Z_N = U_n\Sigma_n V_n^T$$

State sequence is hence estimated as,

$$\hat{X}_{s,N} = \Sigma_n^{1/2} V_n^T$$

We now use this state sequence to find our State Matrices by solving the linear least squares problem:

$$\| \begin{bmatrix} \hat{X}_{s+1,N} \\ \hat{Y}_{s,1,N-1} \end{bmatrix} - \begin{bmatrix} A_T \\ C_T \end{bmatrix} \begin{bmatrix} \hat{X}_{s,N-1} \end{bmatrix} \|_F^2$$

We solve this problem analytically, using the general least squares solution to a problem $\|y - Fx\|_2^2$ which is $\hat{x} = (F^T F)^{-1} F^T y$.
This completes the state estimation and computing the state matrices.

Question 2

Using the results from the previous question, we can compute the co-variances of our state estimate and identification data (including cross-covariance). This is done using the residuals from our optimal state estimate and our identified state matrices:

$$\begin{bmatrix} \hat{W}_{s,1,N-1} \\ \hat{V}_{s,1,N-1} \end{bmatrix} = \begin{bmatrix} \hat{X}_{s+1,N} \\ \hat{Y}_{s,1,N-1} \end{bmatrix} - \begin{bmatrix} A_T \\ C_T \end{bmatrix} \begin{bmatrix} \hat{X}_{s,N-1} \end{bmatrix}$$

Thus, our co-variances are computed as:

$$\begin{bmatrix} \hat{Q} & \hat{S} \\ \hat{S}^T & \hat{R} \end{bmatrix} = \underset{N\to\infty}{Lt} \frac{1}{N} \begin{bmatrix} \hat{W}_{s,1,N-1} \\ \hat{V}_{s,1,N-1} \end{bmatrix} \begin{bmatrix} \hat{W}_{s,1,N-1}^T & \hat{V}_{s,1,N-1}^T \end{bmatrix}$$

Now that we have our co-variances, we can use the solution of the stationary Kalman equation to find optimal kalman gain K_s. This is done by solving a Ricatti equation of the form:

$$\hat{P} = \hat{A}_T \hat{P} \hat{A}_T^T + \hat{Q} - (\hat{S} + \hat{A}_T \hat{P} \hat{C}_T^T)(\hat{C}_T \hat{P} \hat{C}_T^T + \hat{R})^{-1} (\hat{S} + \hat{A}_T \hat{P} \hat{C}_T^T)^T$$

And the Kalman Gain is:

$$\hat{K}_T = (\hat{S} + \hat{A}_T \hat{P} \hat{C}_T^T)(\hat{C}_T \hat{P} \hat{C}_T^T + \hat{R})^{-1}$$

This completes the computation of the Covariances and the Kalman Gain.

Question 3

Now that we know our State Space matrices (lets call them A_s, C_s) and our optimal Kalman Gain (K_s), we can use our turbulence model to obtain optimal prediction for our next state i.e. $\hat{x}(k+1|k)$ and consequently the optimal predition for the next open loop slope $\hat{s}(k+1|k)$. If the current output slope is $s(k)$, we can find the current innovation signal and hence write our prediction $\hat{x}(k+1|k)$ as:

$$\hat{x}(k+1|k) = A_s \hat{x}(k|k-1) + K_s(s(k) - C_s \hat{x}(k|k-1)) \tag{12}$$

Starting from an initial estimate $x(0)$, our Kalman implementation should eventually converse and provide a good estimate of $\hat{x}(k+1|k)$ and in turn, of $\hat{s}(k+1|k)$:

$$\hat{s}(k+1|k) = C_s \hat{x}(k+1|k)$$

Since we need the optimal prediction of ϕ, we can now use $\hat{s}(k+1|k)$ to find $\hat{\phi}(k+1|k)$ using the expression derived in Part 1.1 which is:

$$\hat{\phi}(k+1|k) = V_1 \Sigma^{-1} U_1^T \hat{s}(k+1|k)$$

where $V_1 \Sigma^{-1} U_1^T$ represents the pseudo inverse of the matrix G found using SVD.
We can now use this $\hat{\phi}(k+1|k)$ to arrive at an expression for the residual $\hat{\epsilon}(k+1|k)$:

$$\hat{\epsilon}(k+1|k) = \hat{\phi}(k+1|k) - Hu(k)$$
$$\hat{\epsilon}(k+1|k) = V_1 \Sigma^{-1} U_1^T \hat{s}(k+1|k) - Hu(k)$$

So finally we get:

$$\hat{\epsilon}(k+1|k) = V_1 \Sigma^{-1} U_1^T C_s \hat{x}(k+1|k) - Hu(k) \tag{13}$$

Question 4

The control action that minimises $\hat{\epsilon}(k+1|k)$ as in equation (13) can be found by solving the least squares problem:

$$\|\hat{\epsilon}(k+1|k)\|_2^2 = \|V_1\Sigma^{-1}U_1^T C_s\hat{x}(k+1|k) - Hu(k)\|_2^2$$

The analytical solution looks like:

$$\hat{u}(k) = (H^T H)^{-1}H^T V_1\Sigma^{-1}U_1^T C_s\hat{x}(k+1|k) \tag{14}$$

Hence we have found a minimising control input u.

Question 5

Approach: We ran the subspace Identification and Validation for the given datasets ϕ_{ident} and ϕ_{sim} respectively.
The System Identification cycle was completed multiple times for different sets of ϕ_{ident} and ϕ_{sim} to avoid overfitting any particular dataset and to arrive at reliable average variance and VAF values.
Matlab Implementation attached in the Appendix.

Question 6

Results:

- **Average VAF** for the estimate $\hat{\phi}(k+1|k)$ achieved through this method is **90.8295%**. This clearly shows that the state space turbulence model, found using the Subspace Identification method, works best as compared to models in Part 1 and 2 in terms of estimating the actual wavefront ϕ.
- **Average Variance** of the residual achieved through this method is **2.0329**. This displays the improved closed loop performance of our method used in Part 3 as compared to Part 1 and 2.
- The parameter n for Subspace Identification was chosen by observing the Singular Values of the RQ approximation $R_{21}R_{11}^{-1}Z_N$ since this SVD has the same column space as the Observability matrix O_s. The plot for the Singular values is shown in Figure 1 below:

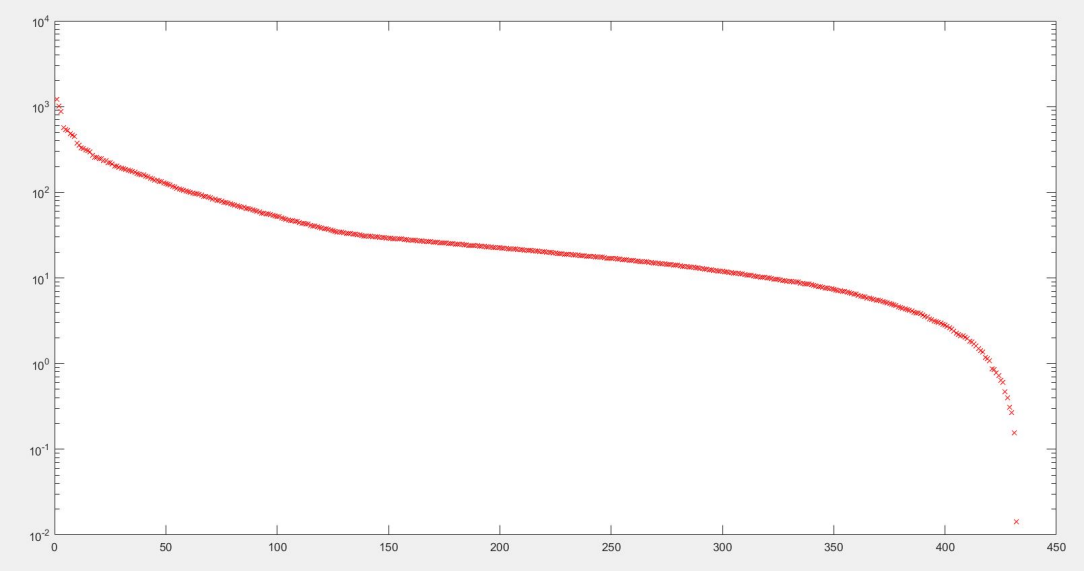

Figure 1: Singular Value Distribution of $R_{21}R_{11}^{-1}Z_N$

An analysis of this distribution shows that the first 3 to 6 singular values are the most important and subsequent Singular values are quite close to each other in terms of magnitude. Thus, a mimimum order of 6 is essential. While there are improvements on choosing a larger order n, we get diminishing returns. Another consideration for n is the amount of computation that would be required for the turbulence model identification and validation when choosing a large n.
After experimenting with different identification datasets and validating them, we arrived at an optimum value of $n = 60$. This captures most of the information from the dataset while not being too large for computation and achieves good results for Variance and VAF

- The parameter s for Subspace Identification was chosen based on the law: $2p^2s > n$, where p is the number of lenslets in the Shack-Hartmann sensor and so $(p+1)^2$ is the dimension of a wavefront ϕ. In principle, this condition ensures that s is large enough to identify an n^{th} order system and satisfies rank conditions.
  After experimenting with different identification datasets and validating them, we arrived at an optimum value of $s = 6$. We noticed that using a large value of s i.e. a value close to n leads to worse performance. Hence s was kept small while still satisfying the law: $2p^2s > n$.

- The parameters N_{val} and N_{id} were not used since we performed our validation on the simulation dataset ϕ_{sim} in the 'AOlooSID' function.

4. CRITICAL THINKING

Question 1

In this question, we make use of a slightly different version of the random walk model presented in Section 1, with the control law requiring the minimization of the residual slopes s(k+1) instead of $\epsilon(k+1)$.

We know from equation (5) of the assignment that:

$$\hat{s}(k+1|k) = G\hat{\epsilon}(k+1|k)$$

Writing $\hat{\epsilon}(k+1|k) = \hat{\phi}(k+1|k) - Hu(k)$, we obtain the relation:

$$\hat{s}(k+1|k) = G\hat{\phi}(k+1|k) - GHu(k)$$

Since we are making use of the random walk model as described by equation (9), we know the relation: $\hat{\phi}(k+1|k) = \hat{\phi}(k|k)$

Moreover, we also know from **??** the estimate for $\hat{\epsilon}(k|k)$, which can be written as $\hat{\phi}(k|k) - Hu(k-1)$. Thus, we get the estimate for $\hat{\phi}(k|k)$ in terms of s(k), u(k), and u(k-1).

Combining the above pieces of information together, we get the following expression for $\hat{s}(k+1|k)$:

$$\hat{s}(k+1|k) = G\left(C_\phi(0)G^T(GC_\phi(0)G^T + \sigma_e^2 I)^{-1}s(k) + Hu(k-1)\right) - GHu(k)$$

We clearly see that $\underset{u(k)}{min}\|\hat{s}(k+1|k)\|_2^2$ is a linear least-squares problem if we know the vector u(k-1).

We see that G*H produces a matrix that is not full-rank, and hence we follow the approach from Section 1's Question 1, in which there is no assumption made on the rank.

We get the optimum value of u(k),

$$\hat{u}(k) = V_1\Sigma^{-1}U_1^T G\left(C_\phi(0)G^T(GC_\phi(0)G^T + \sigma_e^2 I)^{-1}s(k) + Hu(k-1)\right)$$

Where U_1, V_1, and Σ are the matrices obtained on performing an SVD of the G*H matrix considering the linearly independent vectors and their singular values.

We know that u(0) = 0, since we do not apply any actuation initially. Based on this information, we calculate the value for u(1) by solving the above minimization problem. Subsequently, we solve the minimization problems for all u(k) values.

Thus, we calculate the variance of $\epsilon(k)$ based on the values calculated from the above u(k) values and the $\phi_{sim}(k+1)$ values provided to us.

We obtain approximately the same values as in Question 1, that is, a variance of ~ 6.5 and a VAF of ~ 70%

Question 2

The following values are obtained for the different methods:

- Random walk model: var(ϵ) ~ 6.5, and VAF ~ 70 %
- VAR1 model: var(ϵ) ~ 3.5, and VAF ~ 83 %
- Subspace Identification model: var(ϵ) ~ 2.1, and VAF ~ 90 %
- Slope-minimizing model: var(ϵ ~ 6.5, and VAF ~ 70 %

The above results follow a trend which is expected, since the models used greatly affect the results obtained.

In the **random walk models**, since we do not make enough use of the data provided to us and follow an approach in which we assume the wavefront to be changing based on a random turbulence value, we cannot really predict accurately the one step ahead wavefronts and thus end up getting large variance values and low VAF values. Moreover, for both the slope-minimizing and the Question 1 RW model, we get approximately the same values since the underlying model is the same in both cases and the minimization of either the predicted s(k+1) values or $\epsilon(k+1)$ should yield the same results (we do not make use of any further information in the slop-minimizing model and thus cannot expect better results).

In the **VAR1 model**, we make use of a stationary Kalman filter (constructed using the data available to us) and thus can expect better values of variance and VAF than the random walk models (the observer provides a correction term to the model through the Kalman filter, based on which the predicted values are closer to the actual values). We see that the values we obtain are indeed in line with our expectations.

In the **subspace identification model**, we make use of all the data we have about the system's output and input data, and construct models of different orders that best represent the system based on the data. After choosing an order based on training data, validation data is used to test the constructed system. Hence, we clearly see that a lot more data is used in this approach to construct our system model, and given that the validation data is from the same family of datasets, we can expect our constructed system model to be quite accurate

when tested with the validation data. Moreover, there is no prior assumption made on the system other than that it is represented by a stochastic state-space model. We clearly see from the results that the subspace identification model performs the best in terms of the performance parameters, as expected.

Question 3

The second unobservable mode is one in which the wavefront happens to lie in the nullspace of the G matrix (that is, if it does, the output slope vector is a zero vector which is not truly indicative of the turbulent wavefront). On performing an *imagesc* run on the null vectors, we get the image pattern as seen in Figure 2

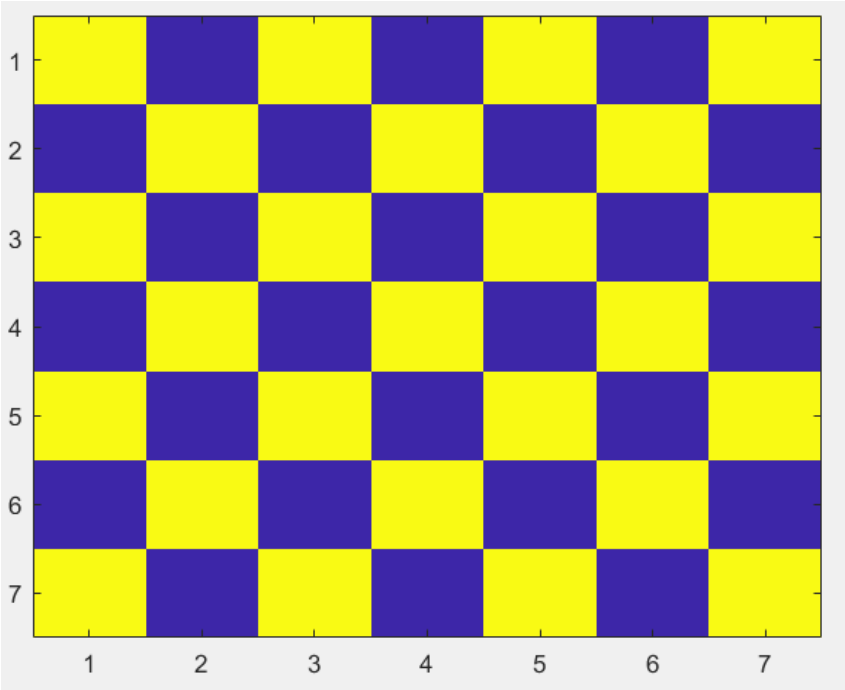

Figure 2: Imagesc representing the kernel of G

Since we make use of wavefront prediction in our models, if we see the above pattern as the predicted wavefront, we can be cautious and change the input (ϕ_{DM}) so as to move away from the wavefront that would otherwise cause the slope vector to be falsely 0.

Question 4

The σ_e term affects the variance of our estimate $\hat{\phi}$.
When σ_e is unknown, we could, in principal, keep σ_e very small and obtain an estimate with

a low variance. However, this estimate of ours would then not fit the actual data very well. We would get a large residual data fitting error.
Alternatively, if σ_e is too large, we overcompensate for the variance of the actual data and our estimate is again sub-par.
So, σ_e has to be tuned to closely approximate the variance of the actual data's noise. This can be done by Data-preprocessing to estimate the variance of the noise in the actual data.

Question 5

The control law we derived in Section 1 is represented by equation 3, in which we calculate the input to be applied at a time instant based on the slope measurement at that time instant and the previous input.

It is a recursive relation, and we assume that at the first time instant, we do not apply any control input. Based on this, we calculate the control input ($\phi_{DM}(k)$) for the rest of the time instants based on the control law represented by equation (6) in the assignment (on which equation 3's derivation is based).

We see that in

$$\hat{\delta} u(k) = (H^T H)^{-1} H^T C_\phi(0) G^T (G C_\phi(0) G^T + \sigma_e^2 I)^{-1} s(k)$$

there is a mix of online and offline computations, as listed below:

- say, V = $(H^T H)^{-1} H^T C_\phi(0) G^T (G C_\phi(0) G^T + \sigma_e^2 I)^{-1}$. We know that V can be computed offline since its value does not depend on any sample instant k (the H, $C_\phi(0)$ and G matrices do depend on the sensor array size, but since the above computation needs to be done only once, there's no exponential increase in the number of computations with increasing loop size)

- Multiplying V with s(k) cannot be done offline since it depends on the slope vector s(k) at time instant k, and the number of computations grows quadratically with increasing sensor array size (since there is a squaring term involved).

- Likewise, the number of computations for calculating u(k) from u(k-1) grows quadratically with the increasing actuator array size. Since it is a recursive relation, it cannot be done offline.

Per iteration, there are $2p^2xm^2$ elementary multiplication operations and $(2p^2-1)xm^2$ elementary addition operations from the V*s(k) matrix multiplication, and a further m^2 elementary addition operations from adding the product to u(k-1).

Hence, we have a total of $2p^2xm^2$ multiplication and $2p^2xm^2$ addition operations as online computations.

Thus, we can conclude that this method is not very scalable purely in terms of the sensor and actuator array sizes involved, since with large array sizes, the number of computations per iteration grows quadratically ($\mathcal{O}(n^2)$).

REFERENCES

M. Verhaegen, V. Verdult. "Filtering and System Identification: A Least-Squares Approach", Cambridge University Press, 2007.

APPENDIX

Please find henceforth all MATLAB code associated with this assignment.

Table of Contents


```
% Filtering and Identification - final assignment
% Section 1 - Random Walk Model

close all;

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

Question 1

```
% We see that we have been provided data corresponding to s0(k) and G,
 and
% we have to determine the value of phi(k) such that the error, e(k),
 is
% minimum, given no prior information. This is clearly a linear least
% squares problem.

rank_G = rank(G);

% We get the rank of G as 47, whereas the number of columns in G is 49
 and
% the number of rows is 72. Hence, G is not full rank, we conclude.

% Getting the SVD of G:
[U, S, V] = svd(G, 0);
```

Question 5

```
% In Question 5, we need to know the rank of the system matrix H so as
 to
% determine a solution for the linear least squares problem minimizing
 the
% 2-norm of epsilon (k + 1|k)
rank_H = rank(H);

% We get the rank of H as 49, and hence we conclude that H is a full
 rank,
% invertible matrix.
```

Question 6

```
% The function AOloopRW is present in the same directory as the
 current
% one.

% Considering the first cell in the given cell array
phiSim_1 = phiSim{1,1};

% Calculating the dimensions of each phiSim cell
[phi_len, n] = size(phiSim_1);

% Covariance matrix is square
covariance_phi = zeros(phi_len, phi_len);

% We are provided with a cell array of 20 datasets for phiSim
num_Datasets = length(phiSim);

variance_closed_loop = 0;
variance_no_control = 0;

for cellIndex = 1:num_Datasets
    phi_currentCell = phiSim{1,cellIndex};
    for k = 1:n
        covariance_phi = covariance_phi +
 (phi_currentCell(:,k)*phi_currentCell(:,k)');
    end

    covariance_phi = covariance_phi/n;

    variance_closed_loop = variance_closed_loop + AOloopRW(G,H,
 covariance_phi, sigmae, phi_currentCell);
    variance_no_control = variance_no_control +
 AOloop_nocontrol(phi_currentCell,sigmae,H,G);
end

% Taking the average of the values obtained from all of the provided
% datasets
variance_closed_loop = variance_closed_loop/num_Datasets;
variance_no_control = variance_no_control/num_Datasets;
```

Question 7

```
VAF_cumulative = 0;

for cellIndex = 1:num_Datasets
    phi_currentCell = phiSim{1,cellIndex};
    covariance_phi = zeros(phi_len, phi_len);
    for k = 1:n
        covariance_phi = covariance_phi +
 (phi_currentCell(:,k)*phi_currentCell(:,k)');
    end
```

```
    covariance_phi = covariance_phi/n;

    VAF_cumulative = VAF_cumulative + VAF_RW(G,H, covariance_phi,
 sigmae, phi_currentCell);
end

VAF = VAF_cumulative/num_Datasets;
```

Published with MATLAB® R2018a

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

VAF calculation function for the random walk model

```
function [VAF] = VAF_RW(G,H, covariance_phi, sigma_e, phi_sim)
% VAF calculation
% 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
% VAF : VAF value

% 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);

% The following term is multiplied with the slope vector to obtain the
 eps_hat(k|k)
% value
eps_pred_multiplier_matrix = 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);

% 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));

% Calculation of variance accounted for:

deviation_norm_sum = 0;
actual_phi_norm_sum = 0;

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

    predicted_phi = H*eps_pred_multiplier_matrix*slope_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));

end
```

Published with MATLAB® R2018a

```
% Filtering and Identification - final assignment
% Section 2 - VAR model

close all;

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

phi_currentCell = phiSim{1,1};
[phi_len, n] = size(phi_currentCell);

% Total number of given datasets with simulated phi values
num_Datasets = length(phiSim);
```

Question 7

```
var_KalmanAR = 0;
VAF_Kalman = 0;

for cellIndex = 1:num_Datasets
    phi_currentCell = phiSim{1,cellIndex};

    % Given definition for C_phi(0)
    covariance_phi_0 = zeros(phi_len, phi_len);
    for k = 1:n
        covariance_phi_0 = covariance_phi_0 +
 (phi_currentCell(:,k)*phi_currentCell(:,k)');
    end

    covariance_phi_0 = covariance_phi_0/n;

    covariance_phi_1 = zeros(phi_len, phi_len);

    % Given definition for C_phi(1)
    for k = 2:n
        covariance_phi_1 = covariance_phi_1 +
 (phi_currentCell(:,k)*phi_currentCell(:,k-1)');
    end

    covariance_phi_1 = covariance_phi_1/(n-1);

    % Computing the values for the Kalman filter
    [A, Cw, K] = computeKalmanAR(covariance_phi_0, covariance_phi_1,
 G, sigmae);
    % Computing the variance for the current dataset
    var_KalmanAR = var_KalmanAR + AOloopAR(G,H, covariance_phi_0,
 sigmae, A, Cw, K, phi_currentCell);

    %Computing the VAF values
    VAF_Kalman = VAF_Kalman + VAF_VAR(G,H, sigmae, A, K,
 phi_currentCell);
end
```

```
% Taking the average values over 20 datasets
var_KalmanAR = var_KalmanAR/num_Datasets;
VAF_Kalman = VAF_Kalman/num_Datasets;
```

Question 8

```
VAF_Kalman
```

Published with MATLAB® R2018a

computeKalmanAR function

```
function [A, Cw, K] = computeKalmanAR(C_phi_0, C_phi_1, G, sigma_e)

%COMPUTEKALMANAR Computes parameters for the Kalman filter in the VAR
 model
%   IN
% C_phi_0 : The auto-covariance matrix for the given phi dataset
% C_phi_1 : The C_phi(1) matrix for the given phi dataset
% G : The G matrix relating epsilon(k) to s(k)
% sigma_e : SD of the noise vector

% OUT
% A : State matrix defining the relation between the current and next
% states
% Cw : Covariance matrix for process noise
% K : Stationary Kalman filter gain

phi_len = size(C_phi_0, 1);
A = C_phi_1/C_phi_0;

Cw = (eye(size(A)) - A*A)*C_phi_0;

% For the DARE, we need the Q matrix to be symmetric. So we take the
% average of the Cw matrix and its transpose
Cw = (Cw + Cw')/2;

R = (sigma_e^2)*eye(size(G,1));

E = eye(phi_len);
S = zeros(size(G'));

% Making use of the DARE provided by MATLAB to compute the covariance
% matrix for epsilon
P = dare(A',G',Cw,R,S,E);

% Kalman gain
K = A*P*G'/(G*P*G' + R);

end
```

Published with MATLAB® R2018a

AOLoopAR function

```
function [var_eps] = AOloopAR(G,H, covariance_phi, sigma_e, A, Cw, K,
 phi_sim)
% Variance calculation of an AO system in the closed-loop
 configuration
% with a Kalman filter involved
% 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
% A : A matrix from the state-space model
% Cw : Covariance matrix of the process noise
% K : Stationary Kalman filter gain
% 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);

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

eps_kplus1k = 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.

% s(k) = G*eps(k) + e(k)
% Hence s(k) = G*phi(k) - G*H*u(k-1) + e(k)

% u(k) = 0 for k < 1

% eps(k+1|k) = Ks(k) + A*H*u(k-1) - H*u(k) + (A - KG)eps(k|k-1)
```

```
% eps(k+1|k) = K*G*phi(k) - K*G*H*u(k-1) + K*e(k) + A*H*u(k-1) -
 H*u(k) + (A - KG)eps(k|k-1)

% First slope value, no input applied before
s_k = G*phi_sim(:,1) + sigma_e*randn(n_G,1);

% Initial input
u(:,1) = H\(K*s_k);

% eps(1|0) = 0;
%eps_kplus1k = eps(k+1|k)
eps_kplus1k(:,1) = K*G*phi_sim(:,1) + K*sigma_e*randn(n_G,1) -
 H*u(:,1);

% Initial eps(k)
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
    % Slope value
    s_k = G*phi_sim(:,k) - G*H*u(:,k-1) + sigma_e*randn(n_G,1);

    % Optimum input value
    u(:,k) = H\(K*s_k + (A - K*G)*eps_kplus1k(:,k-1) + A*H*u(:,k-1));

    % Next eps(k+1|k) value
    eps_kplus1k(:,k) = K*s_k + A*H*u(:,k-1) - H*u(:,k) + (A -
 K*G)*eps_kplus1k(:,k-1);

    % eps(k) value = phi(k) - Hu(k-1)
    eps_k(:,k) = phi_sim(:,k) - H*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

VAF calculation function for the VAR model

```
function [VAF] = VAF_VAR(G,H, sigma_e, A, K, phi_sim)
% VAF calculation
% IN
% G     : measurement matrix
% H     : influence matrix mapping the wavefront on the mirror
% sigma_e : measurement noise parameter for determining its covariance
% A : A matrix from the state space model
% K : Kalman gain value
% phi_sim : simulation data for the wavefront
% OUT
% VAF : VAF value

% 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);

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

% An assumption is that H is full rank.

% s(k) = G*eps(k) + e(k)
% Hence s(k) = G*phi(k) - G*H*u(k-1) + e(k)

% u(k) = 0 for k < 1

% eps(k+1|k) = Ks(k) + A*H*u(k-1) - H*u(k) + (A - KG)eps(k|k-1)

% eps(k+1|k) = K*G*phi(k) - K*G*H*u(k-1) + K*e(k) + A*H*u(k-1) -
 H*u(k) + (A - KG)eps(k|k-1)

% First slope value, no input applied before
s_k = G*phi_sim(:,1) + sigma_e*randn(n_G,1);

% Initial input
u(:,1) = H\(K*s_k);

% e(1|0) = 0;
eps_kplus1k(:,1) = K*G*phi_sim(:,1) + K*sigma_e*randn(n_G,1) -
 H*u(:,1);

deviation_phi_norm_sum = 0;
actual_phi_norm_sum = 0;
```

```
for k = 2:T-1
    % Slope value
    s_k = G*phi_sim(:,k) - G*H*u(:,k-1) + sigma_e*randn(n_G,1);

    % Optimum input value
    u(:,k) = H\(K*s_k + (A - K*G)*eps_kplus1k(:,k-1) + A*H*u(:,k-1));

    % Next eps(k+1|k) value
    eps_kplus1k(:,k) = K*s_k + A*H*u(:,k-1) - H*u(:,k) + (A -
 K*G)*eps_kplus1k(:,k-1);

    predicted_phi = eps_kplus1k(:,k) + H*u(:,k);
    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_phi_norm_sum = deviation_phi_norm_sum + current_norm;
    actual_phi_norm_sum = actual_phi_norm_sum +
 (norm(phi_sim_mean_removed))^2;

end

mean_deviation_norm = deviation_phi_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

Table of Contents


```
% Filtering and Identification - final assignment
% Section 3 - SI State Space Model
%

close all;

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

Question 5: Sub Identification

```
% The functions SubId and AOloopSID is present in the same directory
 as the current
% one.

% Considering the first cell in the given cell array
phiIdent_1 = phiIdent{1,20}; % Trials with different identification
 datasets

% Calculating the dimensions of each phiSim cell
[phi_len, n] = size(phiIdent_1);

% Size of G matrix
n_G = size(G,1);

% Number of sample points for phi_sim
T = length(phiIdent_1);

% Get sid matrix
sid = G*phiIdent_1 + sigmae*randn(n_G,T);

% Choose s and n paramters
% Chosen using by analysing the SVD using: semilogy(diag(Sigma),'xr');
s = 6;
n = 60;

[As, Cs, Ks] = SubId(sid, s,n); % (Trials with different
 identification datasets)
```

Question 5: Validation

```
phiSim_2 = phiSim{1,6}; % Trials with different simulation datasets
```

```
[var_eps, VAF] = AOloopSID(G,H,As,Cs,Ks,sigmae,phiSim_2);
```

Question 5: Validation over entire dataset

```
% We are provided with a cell array of 20 datasets for phiSim
num_Datasets = length(phiSim);

var_eps_total = 0;
VAF_cumulative = 0;

for cellIndex = 1:num_Datasets
    phi_currentCell = phiSim{1,cellIndex};
    [var_currentCell, VAF_currentCell] =
 AOloopSID(G,H,As,Cs,Ks,sigmae,phi_currentCell);
    var_eps_total = var_eps_total + var_currentCell;
    VAF_cumulative = VAF_cumulative + VAF_currentCell;
end

% Taking the average of the values obtained from all of the provided
% datasets
fprintf("Results:\n");
Variance_avg = var_eps_total/num_Datasets
VAF_avg = VAF_cumulative/num_Datasets
```

Published with MATLAB® R2018a

SubId function

```
function [As, Cs, Ks] = SubId(sid,s,n)
% SubID calculations
% IN
% sid : Slopes of Identification data
% s   : Parameter 's'. Rows of Henkel matrix.
% n   : Order of states 'n'
% OUT
% As : State Space matrix
% Cs : State Space matrix
% Ks : Kalman Gain in Observer form

% Number of sample points for Subspace Identification
N = size(sid,2);

% Dimension of sid
M = size(sid,1);

% Construct Hankel Matrix using sid
Y = [];
sid_stacked = sid(:);

for i=1:N-s+1
    Y = [Y sid_stacked(M*(i-1)+1:M*(s+i-1))];
end

Ypast = Y(:,1:N-2*s+1);
Yfuture = Y(:,s+1:N-s+1);

R = triu(qr([Ypast; Yfuture]'))';
R32 = R(M*s+1:M*2*s,1:M*s);
R22 = R(1:M*s,1:M*s);

[U,Sigma,V] = svd(R32*inv(R22)*Ypast);

% Assigning Sigma for semilogy for choosing n
assignin('base','Sigma',Sigma);

% Reduced System
U = U(:,1:n);
Sigma = Sigma(1:n,1:n);
V = V(:,1:n);

X = sqrt(Sigma)*V';

% Solve least squares problem for X
Xpast = X(:,1:end-1);
Xfuture = X(:,2:end);
Ylsq = sid(:,s+1:end-s);

F = Xpast;
```

```
Sys = (F*(F'))\(F*([Xfuture; Ylsq]'));
Sys = Sys';

As = Sys(1:n,:);
Cs = Sys(n+1:end,:);

% Find K using residuals
Res = [Xfuture; Ylsq] - [As; Cs]*Xpast;
Nt = size(Res,2);
Covar = (Res*Res')/Nt;
Q = Covar(1:n,1:n);
S = Covar(1:n,n+1:end);
R = Covar(n+1:end,n+1:end);

% Choosing P from the solution of the DARE, we can define a Kalman
 gain Ks
% that is asymptotically stable.
% Solution of DARE:
E = eye(n,n);
[P,~,~] = dare(As',Cs',Q,R,S,E);

% Kalman Gain:
Ks = (S + As*P*(Cs'))*inv(R + (Cs*P*(Cs')));

end
```

Published with MATLAB® R2018a

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

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