```
%% DISCLAIMER: This code-file is in developemenet stage.
%%% This code contains t-SNE using MATLAB in-built functions



%%
close all; clc; clear;


%% Load the datasheet
% load octuple_tank_data_28_08_24.mat
load octuple_tank_data_11_11_24.mat



%% 0. Extract your database
%%=======================%%

%%%% T == res
%%% Outputs --> Col 2 to 9
%%% Inputs --> Col 10 to 13
%%% Disturbances --> Col 14 to 15

xt = res(:,2:end);
[M,N] = size(xt);

Dtrain = xt(1:M/2,:);
[m1,n1] = size(Dtrain);

Dtest = xt(1+(M/2):M,:);
[m2,n2] = size(Dtest);



%% 1A. NORMALIZATION OF TRAINING DATASET
%%=======================================%%

xm = mean(Dtrain);
Sdm = std(Dtrain);

Xbar = (Dtrain - xm(ones(m1,1),:)) ./ (Sdm(ones(m1,1),:));
```

```
CV =cov(Xbar);
[U,S,V] = svd(CV);



%% 1B. DIMENSION-REDUCTION OF (NORMALIZED) TRAINING DATASET USING PCA
%%=====================================================================%%

%%% In the file coeff = COEFF and X = SCORE
[COEFF, X1, LATENT, TSQUARED, EXPLAINED, MU] = pca(Xbar); %% LATENT (i.↙
e. EigValued Matix)


% prompt = input('\n\nEnter the percentage of significance (b/w 80 to 95↙
%) = '); %%% self explanatory !!
prompt = 98;

%percent = input(prompt);
percent = prompt/100;

k=0; %%% Initial count = 0

for i = 1:size(LATENT,1)
    alpha(i)=sum(LATENT(1:i))/sum(LATENT);
    if alpha(i)>=percent
            k=i;
            break;
    end
end

% %---------- Display the details of PCs -------------------%

for ij = 1
    fprintf('\n\n\n=================================\n')
    fprintf('|| Octuple-Tank system: Test Results ||\n')
    fprintf('=================================\n\n\n')

    disp('//////// PART A:: SYSTEM INFO ///////'),

    fprintf('\n** Total Number of Observations (Database) = %d \n', M)
```

```
    fprintf('\n** Total Number of Observations (Training) = %d \n', m1)
    fprintf('\n** Total Number of Observations (Testing) = %d \n\n', m2)


    % disp('~~~~~~~~~~~~~~~~')
    % fprintf('Obervations\n')
    % disp('~~~~~~~~~~~~~~~~')

    fprintf('\n\n\n'); disp('//////// PART B:: PCs INFO ///////'),
    fprintf('\n')

    fprintf('\n==> The percentage of significance set for PC↙
contribution = %0.4f', prompt)

    fprintf('\n\n(1) No. of PCs chosen = %d out of 14 I/O vectors.\n',k)

    fprintf('\n(2) Cumul. PC contrib: \n   ********************\n '),↙
alpha,

    TotPCcontr = alpha(1,end) * 100;
    fprintf('## Total PC contribution computed = %0.4f\n', TotPCcontr)

end

princ = X1(:,1:k); %%% upto first kth PCs of Score-matrix X1
% per = LATENT/sum(LATENT);
%
% nnn = [1:length(EXPLAINED)];
% CExpVar = zeros(1,length(nnn));
% for xyz = 1:length(nnn)
%     CExpVar(xyz+1) = CExpVar(xyz) + EXPLAINED(xyz);
% end
% CExpVar(1) = [];




%% 1C. t-SNE MODEL USING THE FIRST 'k' PCs OF TRAINING DATASET
%%=============================================================%%

NumDimensions=2; %% R^(m1 x k) to R^(m1 x 2) space ::: k--> 1st k PCs of↙
```

```
X1 == princ
%--- The 'Perplexity' value cannot be greater than the number of rows of↙
X
[Y_tr,loss1] = tsne(princ,'Algorithm','exact','Distance','euclidean',↙
InitialY=1e-4*randn(m1,NumDimensions),Perplexity=200,Exaggeration=4,↙
LearnRate=200);

figure(1)
subplot(121)
set(gcf, 'WindowState', 'maximized');

% scatter(Y(:,1),Y(:,2),"filled")
scatter(Y_tr(:,1),Y_tr(:,2),40,'MarkerEdgeColor',[0 .5 .5],...
              'MarkerFaceColor',[0 .7 .7],...
              'LineWidth',1.5),
xlabel('Y_1','fontweight','bold'),ylabel('Y_2','fontweight','bold'),
% gscatter(Y(:,1),Y(:,2)),


title('Training Dataset')
grid



%% 2A. t-SNE FOR NORMALIZED NO-FAULT TEST DATASET
%%==================================================%%

SY = scale1(Dtest,xm,Sdm); %% Normalizing Dtest using the mean and STDEV↙
of Dtrain

% CV_test =cov(SY);
% XT1 = SY*COEFF;
% Xp = SY*COEFF(:,1:k)*COEFF(:,1:k)';   % Xp is the estimation of↙
original data using the PCA model, X=Xp+E


NumDimensions=2; %%
[Y_ts0,loss20] = tsne(SY,'Algorithm','exact','Distance','euclidean',↙
InitialY=1e-4*randn(m1,NumDimensions),Perplexity=200,Exaggeration=4,↙
LearnRate=200);
```

```
% figure(2)
subplot(122);

% scatter(Y_ts0(:,1),Y_ts0(:,2),"filled")
scatter(Y_ts0(:,1),Y_ts0(:,2),40,'MarkerEdgeColor',[0 .7 .7],...
              'MarkerFaceColor',[0 .1 .1],...
              'LineWidth',1.5)
hold on;
% xlabel('Y_1','fontweight','bold'),ylabel('Y_2','fontweight','bold'),




%% 2B. INTRODUCE FAULT INTO TESTING DATASET
%%============================================%%

fprintf('\n\n\n\n\n'); disp('//////// PART C:: FAULT DETAILS ///////'),↙
fprintf('\n\n'),


lim = 1000; %%input('\nEnter the instant of fault, lim =  ');
fprintf('\nFault introduced at %d-th observation in Test dataset.', lim)


IndX = 4; %%% Tank# 4
fprintf('\nTank # Selection = %d\n',IndX);

%     %%% FaultID = 'Drift';
%      %%% Ageing slope
%     Bias = 0; Mag_PD =0; idx=0; A=0;
%     Dtest = FDrift(Dtest,lim,IndX,Dslope);
%
%     %%% FaultID = 'Bias';
%       %% How to introduce 10% of total variations ??
%     Dslope = 0; Mag_PD = 0; idx=0; A=0;
%     Dtest = FBias(Dtest,lim,IndX,Bias);
%
%     %%% FaultID = 'Freeze';
%     Bias=0;Dslope =0; Mag_PD =0;
%     [Dtest,idx,A] = FFreeze(Dtest,lim,IndX);
%
```

```
%     %%% FaultID = 'Intermittent';
%     Bias=0;Dslope =0; Mag_PD =0; idx=0; A=0; lim=300;
%     [Dtest] = FIntermit(Dtest,lim,IndX);
%
%     %%% FaultID = 'Precision-Degradation';
%     Dslope = 0.07; %% Enter the slope of Drift fault (w/ Precision↙
Degradation)
%     Mag_PD = 0.3; %% Enter the level of degradation within the↙
interval (0, 1)
%     Bias=0; idx=0; A=0;
%     Dtest = FDPD(Dtest,lim,IndX,Dslope,Mag_PD);
%



%%% ==== Uncomment only those lines that need to be executed ====%%%

for xyz = 1


    % % % -------------------- 1. Bias Fault -------------------------%↙
%
    % FaultID = 'Bias';
    % Bias_value = 5
    % for i = 1:m2
    %     if(i>lim)
    %         Dtest(i,IndX) = Dtest(i,IndX) + Bias_value;
    %     end
    % end


    % -------------------- 2. Drift Fault -------------------------% %
    FaultID = 'Drift';
    Dslope = 0.07;
    r = [];
    for i = 1:(m2-lim)
        r(i)=i;
    end

    for i=1:m2
        if(i>lim)
```

```
            Dtest(i,IndX) = Dtest(i,IndX) + Dslope*r(i-lim);
        end
    end


    % % % ---------------- 3. Drift + Prec. Deg. Fault↙
--------------------% %
    %
    % FaultID = 'Drift+PD';
    % Dslope = 0.09;
    % Mag_PD = 0.45;
    % r = [];
    % for i = 1:(m2-lim)
    %     r(i)=i;
    % end
    %
    % for i=1:m2
    %     if(i>lim)
    %         Dtest(i,IndX) = Dtest(i,IndX) + Dslope*r(i-lim)↙
*Mag_PD*rand(1);
    %     end
    % end



    % % % ------------------- 4. Freeze Fault --------------------% %
    %
    % FaultID = 'Freeze';
    % m2 = size(Dtest,1);
    % % lim = size(Dtest1,1)/2;
    %
    % idx = round(randi(numel(Dtest(lim+1:end, IndX)))/4) %%  Select a↙
random cell index, idx, corr. to MFault_ID vector
    %
    % A = Dtest(lim+idx, IndX); %% Select that cell value corr. to idx
    %
    % for i =1:m2
    %     if(i>lim+idx)
    %         Dtest(i,IndX) = 2*A; %% replace remaining cells as 'A' %%%↙
Bias of 2
    %     end
```

```
    % end


    % % % ------------------- 5. Intermittent Fault ---------------------%↙
%
    %
    % FaultID = 'Intermittent';
    % a= lim + 25;
    % b= lim + 125;
    % c= m2 - 250;
    % d= m2 - 150;
    %
    %
    % for i=1:m2
    %     if (((i > a) && (i < b)) || ((i > c) && (i < d))) %||((i>1050)↙
&& (i<1150)))
    %         Dtest(i,IndX) = Dtest(i,IndX) + 10;
    %     end
    % end

end

%% 2B. NORMALIZATION OF FAULTY TESTING DATASET
%%===============================================%%

fprintf('\n\n\n'); disp('//////// PART D:: PCA MODEL-TEST INFO↙
///////'),
fprintf('\n')


SY = scale1(Dtest,xm,Sdm); %% Normalizing Dtest using the mean and STDEV↙
of Dtrain
CV_test =cov(SY);

% XT1 = SY*COEFF;
Xp = SY*COEFF(:,1:k)*COEFF(:,1:k)';    % Xp is the estimation of original↙
data using the PCA model, X=Xp+E

e = SY - Xp; %% Residual Space
```

```
%% 2C. t-SNE FOR NORMALIZED FAULTY TESTING DATASET
%%=====================================================%%

%%% Xp (by estimation of original data using the PCA model) or SY (by↙
Normalizing Dtest using the mean and STDEV of Dtrain) ??

NumDimensions=2; %%
[Y_ts,loss2] = tsne(SY,'Algorithm','exact','Distance','euclidean',↙
InitialY=1e-4*randn(m1,NumDimensions),Perplexity=200,Exaggeration=4,↙
LearnRate=200);

% figure(2)
subplot(122);

scatter(Y_ts(:,1),Y_ts(:,2),"filled")
xlabel('Y_1','fontweight','bold'),ylabel('Y_2','fontweight','bold'),
% scatter(Y_ts(:,1),Y_ts(:,2),40,'MarkerEdgeColor',[0 .7 .7],...
%                'MarkerFaceColor',[0 .2 .2],...
%                'LineWidth',1.5)
% gscatter(Y(:,1),Y(:,2)),
title(['Test Dataset with ',FaultID,' Fault of value = ', num2str↙
(Dslope)])  %%% 'Test Dataset with Bias Fault = 5'
legend({'No-Fault','w/ Fault'},'Location','best')

grid
sgtitle('t-SNE Models: Training vs. Testing','fontsize',14,'FontName',↙
'Arial','fontweight','bold') %% Global Title




%% 3C. PLOT THE MODELLING LOSSES DURING TESTING vs. TRAINING PHASES
%%===================================================================%%
figure(2)

Loss = [loss1,loss20,loss2];
```

```
set(gcf, 'WindowState', 'maximized');
bar(Loss,'LineWidth',1.2);
title('t-SNE modelling Loss: Training vs Testing')
% subtitle('(b)')
grid




%% 4.

A = inv(princ'*princ)*princ'*Y_ts; %% linear projection matrix A from
high-dimension (princ) to low-dimension, k x 2
```