Create a MLR model

In this section, we first create load the train experiments that were create inthe PCA script and save in the file "Experiment_Train_Set.mat".

These experimental data will be used to create a MLR model to predict final titer, which will serve as term of comparison for the PLS model trained below.

Load Train and Test Sets

Note: the train set has been created in the script "Script01_PCA.mlx". The test set was created in the script "Script00_Process_Characterization.mlx".

```
clear
load("Experiment_Train_Set.mat")
load("Experiment_Test_Set.mat");
```

Create Multiple Linear Regression Model

We first train (fit) a quadratic MLR model to the final titer, using the data loaded above.

We then characterize the performance of the model using ANOVA, where the user can have information about the importance of each coefficient and of the overal model.

```
mdl = stepwiselm(DoE_nondim,f_DoE,'quadratic');
```

```
1. Removing x3^2, FStat = 0.00056897, pValue = 0.98103
2. Removing x2:x3, FStat = 0.16747, pValue = 0.68346
3. Removing x3:x5, FStat = 1.2615, pValue = 0.26469
4. Removing x2^2, FStat = 1.5363, pValue = 0.21871
```

```
anova_table = anova(mdl)
```

anova_table = 17×5 table

	SumSq	DF	MeanSq	F	pValue
1 x1	6.6410e+04	1	6.6410e+04	4.1060	0.0459
2 x2	3.3788e+04	1	3.3788e+04	2.0891	0.1521
3 x3	2.5198e+05	1	2.5198e+05	15.5792	0.0002
4 x4	8.3995e+05	1	8.3995e+05	51.9325	0.0000
5 x5	9.4018e+04	1	9.4018e+04	5.8130	0.0181
6 x1:x2	7.4085e+05	1	7.4085e+05	45.8051	0.0000
7 x1:x3	9.9723e+04	1	9.9723e+04	6.1657	0.0150
8 x1:x4	3.2702e+06	1	3.2702e+06	202.1912	0.0000
9 x1:x5	4.1979e+05	1	4.1979e+05	25.9546	0.0000
10 x2:x4	1.0082e+06	1	1.0082e+06	62.3350	0.0000
11 x2:x5	3.3390e+05	1	3.3390e+05	20.6442	0.0000
12 x3:x4	3.8507e+05	1	3.8507e+05	23.8084	0.0000

	SumSq	DF	MeanSq	F	pValue
13 x4:x5	2.5576e+05	1	2.5576e+05	15.8133	0.0001
14 x1^2	5.5014e+05	1	5.5014e+05	34.0139	0.0000

⋮

```
anova_summary = anova(mdl,'summary')
```

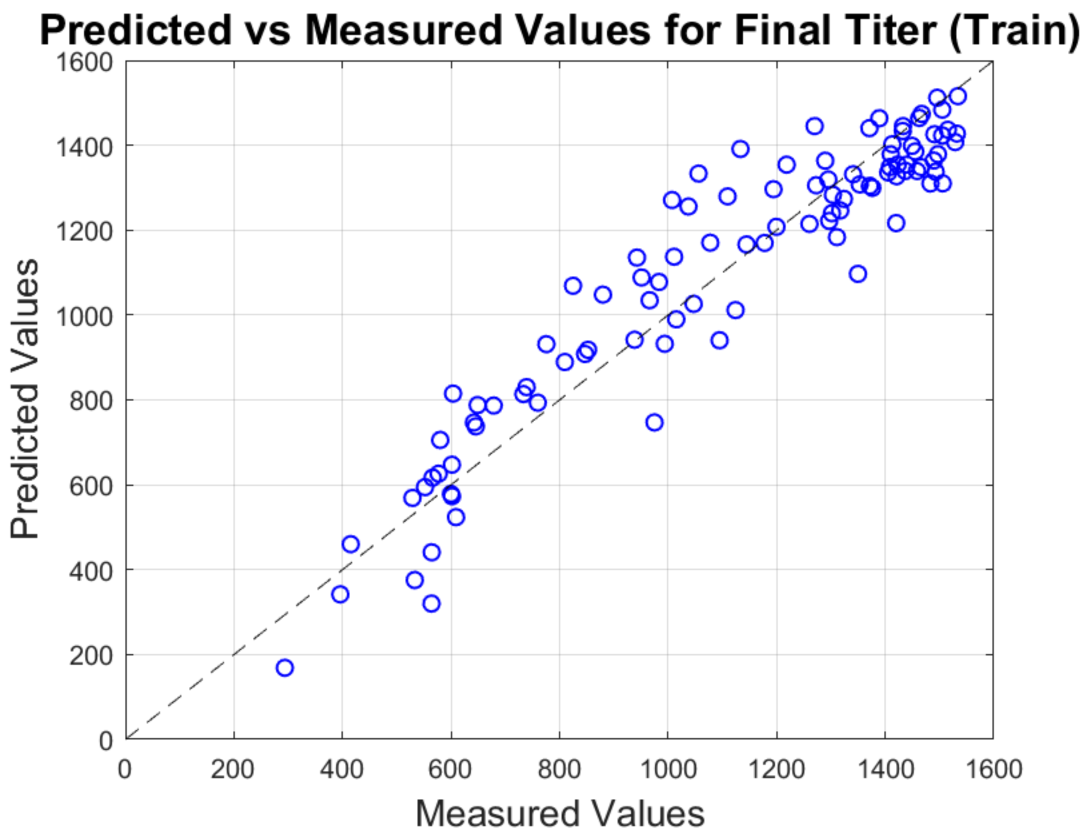
anova_summary = 5×5 table

	SumSq	DF	MeanSq	F	pValue
1 Total	1.1996e+07	99	1.2117e+05	NaN	NaN
2 Model	1.0654e+07	16	6.6585e+05	41.1684	1.1923e-32
3 . Linear	2.1319e+06	5	4.2637e+05	26.3619	7.5112e-16
4 . Nonlinear	8.5218e+06	11	7.7471e+05	47.8986	2.8992e-31
5 Residual	1.3424e+06	83	1.6174e+04	NaN	NaN

Check model prediction on train set

In this section, we check the predictions of the models versus the training data for the final titer.

```
f_DoE_pred = predict(mdl,DoE_nondim);
figure, clf, hold on
plot(f_DoE,f_DoE_pred,'bo','LineWidth',1)
plot([0,1600],[0,1600],'k--')
grid on, box on
title('Predicted vs Measured Values for Final Titer (Train)','FontSize',16)
xlabel('Measured Values','FontSize',14)
ylabel('Predicted Values','FontSize',14)
```




```
abs_RMSE_test = sqrt(sum((f_DoE-f_DoE_pred).^2)/Nruns)
```

```
abs_RMSE_test = 115.8634
```

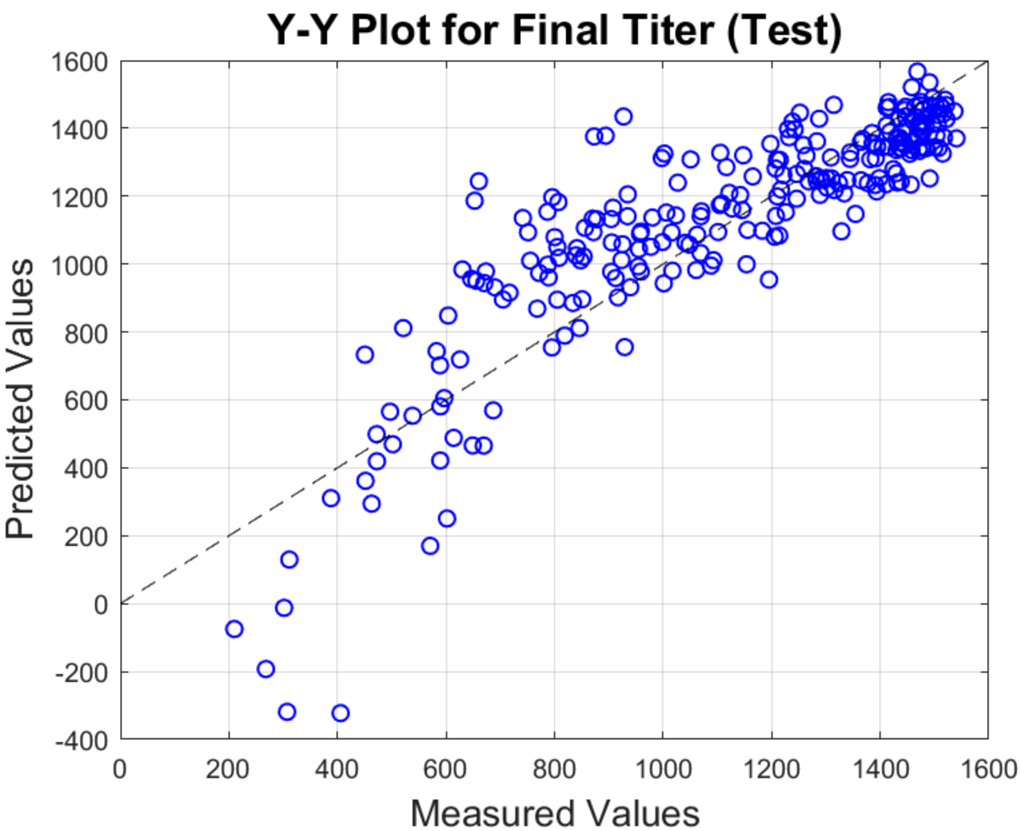
```
rel_RMSE_test = abs_RMSE_test/std(f_DoE_test)
```

```
rel_RMSE_test = 0.3458
```

Check model prediction on test set

In this section, we use the trained model to predict the final titer and we test it on the test set created in the script "Script00_Process_Characterization.mlx".

```
f_DoE_pred = predict(mdl,DoE_LHD_nondim_test);
figure, clf, hold on
plot(f_DoE_test,f_DoE_pred,'bo','LineWidth',1)
plot([0,1600],[0,1600],'k--')
grid on, box on
title('Y-Y Plot for Final Titer (Test)','FontSize',16)
xlabel('Measured Values','FontSize',14)
ylabel('Predicted Values','FontSize',14)
```




```
abs_RMSE_test = sqrt(sum((f_DoE_test-f_DoE_pred).^2)/Nruns_test)
```

```
abs_RMSE_test = 177.8244
```

```
rel_RMSE_test = abs_RMSE_test/std(f_DoE_test)
```

```
rel_RMSE_test = 0.5307
```

Create a PLS1 model

In this section, we train a PLS1 model with the data loaded in the section above.

Create PLS1 model

Create a PLS model to predict the value of the final titer given the initial conditions.

Input matrix: "DoE_nondim". This corresponds to the non-dimensional values of the manipulated variables for each experiment.

```
DoE_nondim
```

```
DoE_nondim = 100×5
   -0.5717   -0.6523    0.9848    0.9737    0.2297
   -0.3024    0.7069    0.8203   -0.0014    0.4837
    0.5457    0.6090   -0.3903   -0.8403    0.7741
   -0.8447   -0.5123   -0.6314   -0.6762   -0.4403
   -0.8349    0.4283   -0.6417   -0.0246   -0.0194
   -0.2885   -0.4566   -0.6663   -0.6938   -0.1485
   -0.4126    0.0496   -0.7736    0.0571    0.9198
```

```
   -0.8874    0.5668    0.4495    0.0238    0.3407
    0.6667    0.4655    0.7369   -0.4457   -0.0778
   -0.7103   -0.3419    0.0907   -0.3811    0.6812
      ⋮
```

Output target: "f_DoE". This corresponds to the final value of titer at the end of each experiment.

```
f_DoE
```

```
f_DoE = 100×1
10^3 ×
    1.3111
    1.0561
    0.9938
    0.4152
    0.8469
    0.6786
    0.9426
    1.1242
    1.4673
    0.8525
      ⋮
```

Select the number of latent variables for the model (the maximum number of latent variables is 5, equal to the number of variables in the input matrix).

```
N_LV = 3;
N_LV = round(N_LV);
if (N_LV > 5) || (N_LV < 2)
    error("Not acceptable input.")
end
```

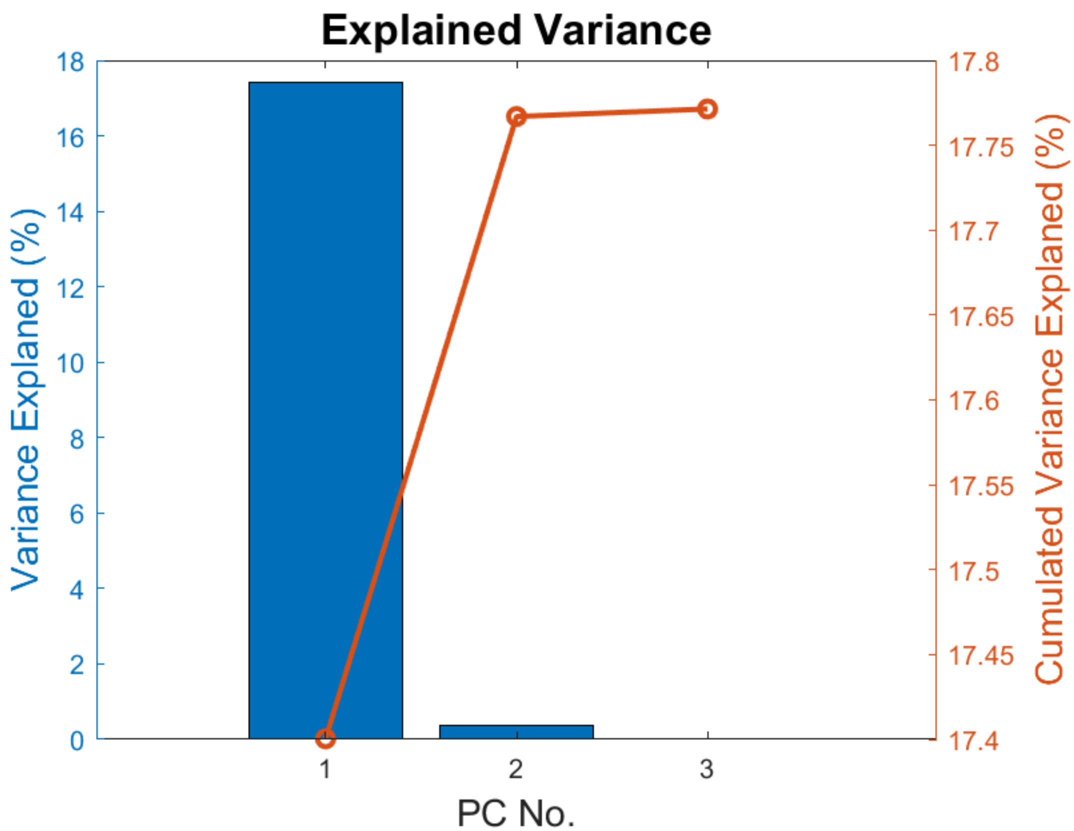
Train the PLS1 model

```
[x_loadings,y_loadings,x_scores,~,beta,pct_var,~,stats] = plsregress(DoE_nondim,f_DoE,N_LV);
```

Analyze the PLS1 model

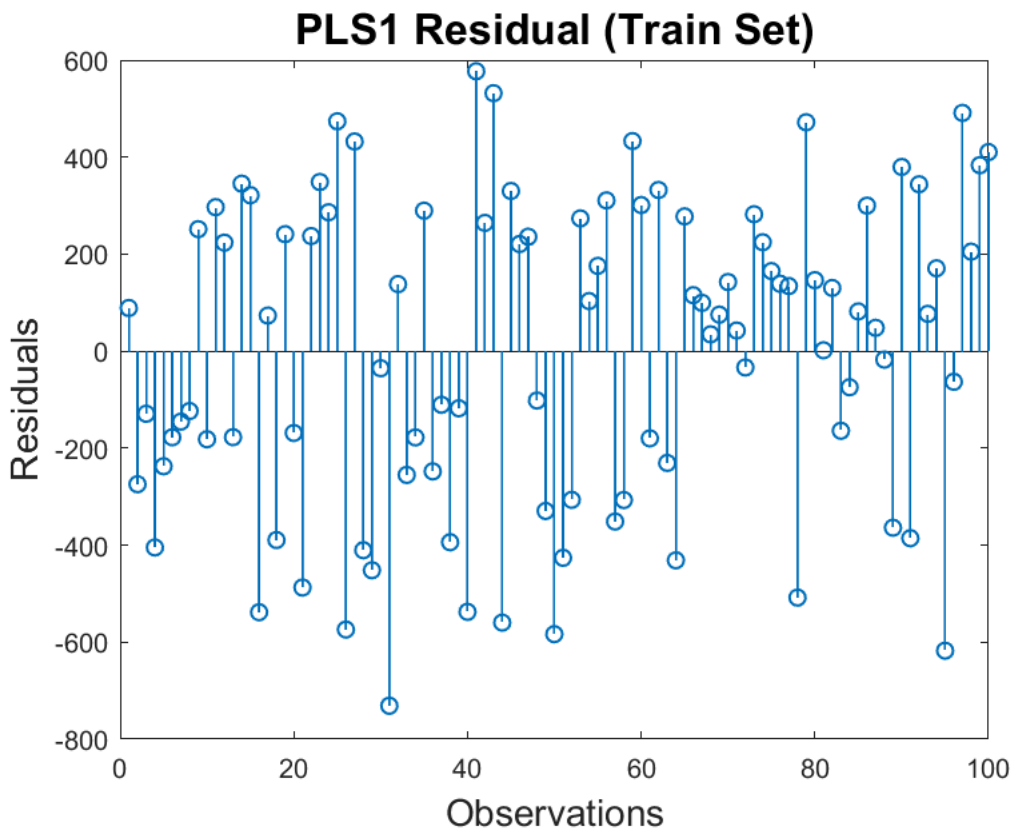
Plot variance explained in the Y

```
figure, clf
yyaxis left
bar(1:N_LV,100*pct_var(2,:))
xlabel('PC No.','FontSize',14),ylabel('Variance Explaned (%)','FontSize',14)
title('Explained Variance','FontSize',16)
yyaxis right
plot(1:N_LV,100*cumsum(pct_var(2,:)),'o-','LineWidth',2,'Color',[0.8500 0.3250 0.0980])
ylabel('Cumulated Variance Explaned (%)','FontSize',14)
```

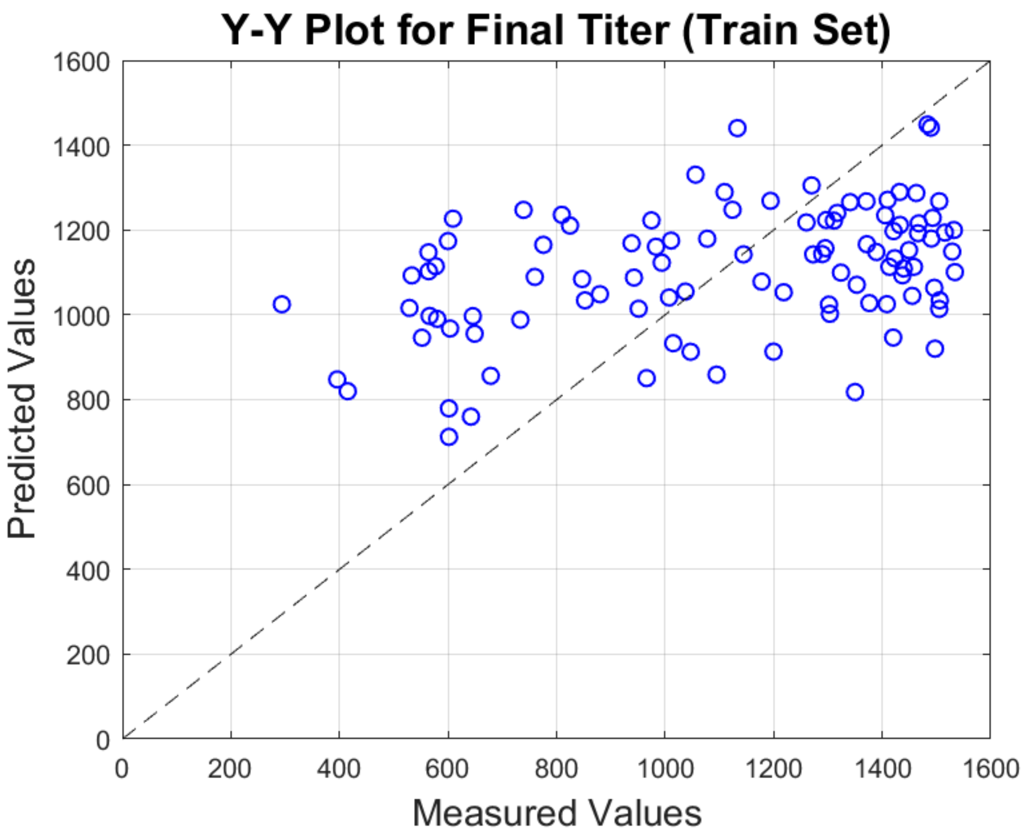


Compute fitted response residuals

```
f_DoE_pred = [ones(Nruns,1),DoE_nondim]*beta;
residuals = f_DoE - f_DoE_pred;
figure, clf
stem(residuals,'LineWidth',1)
xlabel('Observations','FontSize',14), ylabel('Residuals','FontSize',14);
title('PLS1 Residual (Train Set)','FontSize',16)
```




```
figure, clf, hold on
plot(f_DoE,f_DoE_pred,'bo','LineWidth',1)
plot([0,1600],[0,1600],'k--')
grid on, box on
title('Y-Y Plot for Final Titer (Train Set)','FontSize',16)
xlabel('Measured Values','FontSize',14)
ylabel('Predicted Values','FontSize',14)
```




```
SSE = sum(residuals.^2);
RMSE_abs = sqrt(SSE/Nruns)
```

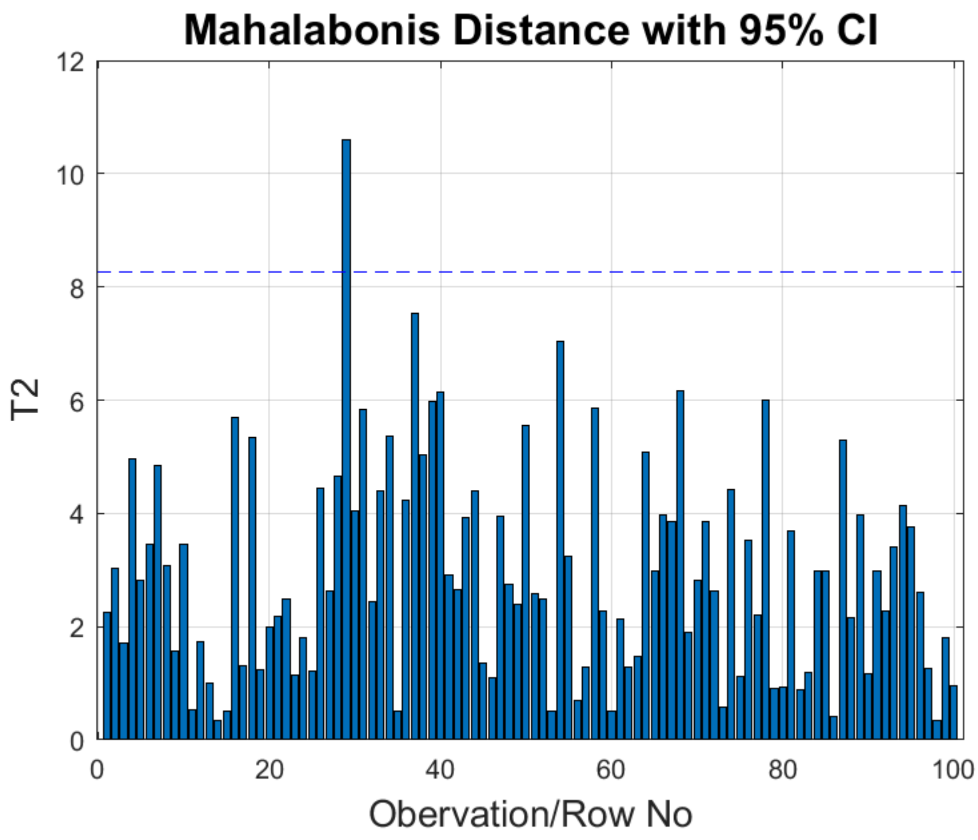
```
RMSE_abs = 314.0734
```

```
RMSE_rel = RMSE_abs/std(f_DoE_test)
```

```
RMSE_rel = 0.9373
```

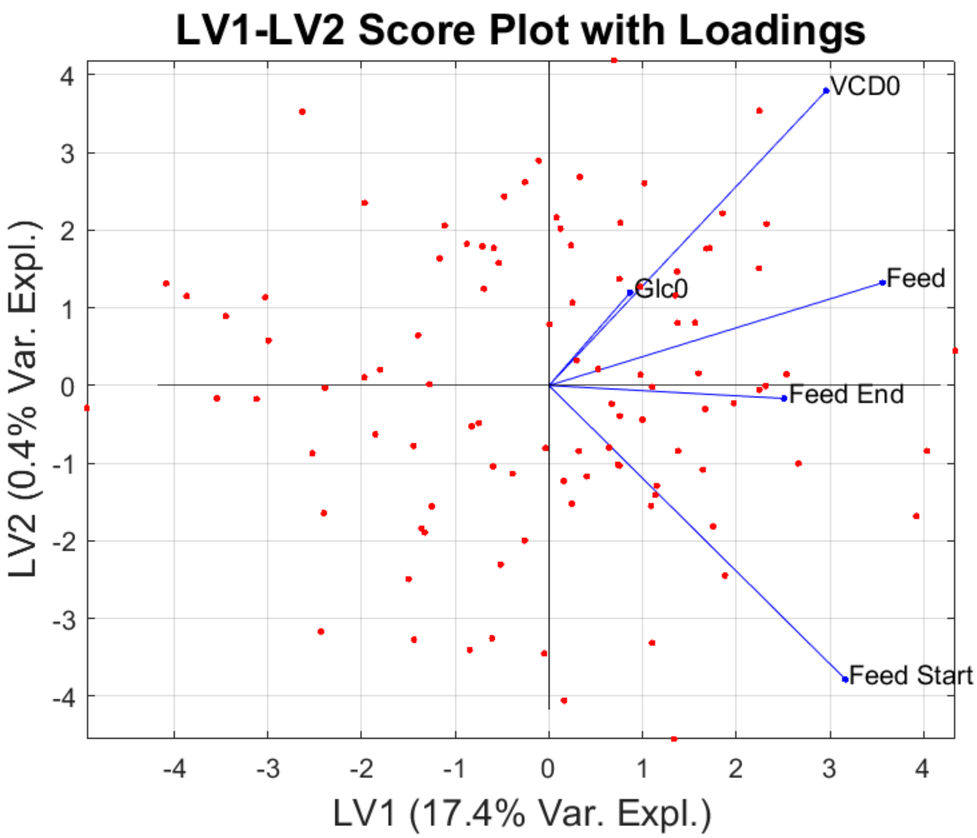
Plot T2

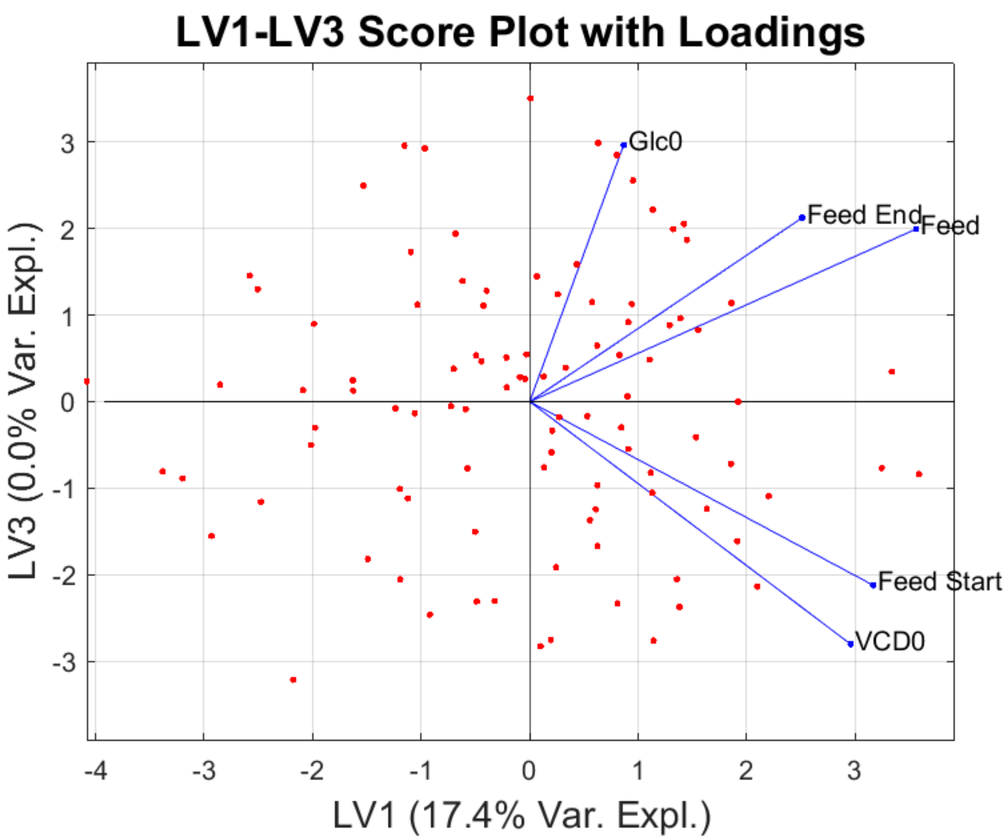
```
alpha = 0.95;
T2 = mahal(x_scores(:,1:N_LV),x_scores(:,1:N_LV));
T2_lim = N_LV*(Nruns-1)/(Nruns-N_LV)*finv(alpha,N_LV,Nruns-N_LV);
figure, clf, hold on
bar(1:Nruns,stats.T2)
plot([0,Nruns+1],[T2_lim,T2_lim],'b--')
box on, grid on
xlabel('Obervation/Row No','FontSize',14),ylabel('T2','FontSize',14)
title('Mahalabonis Distance with 95% CI','FontSize',16)
```



Plot scores (normalized) and loadings for the predictors

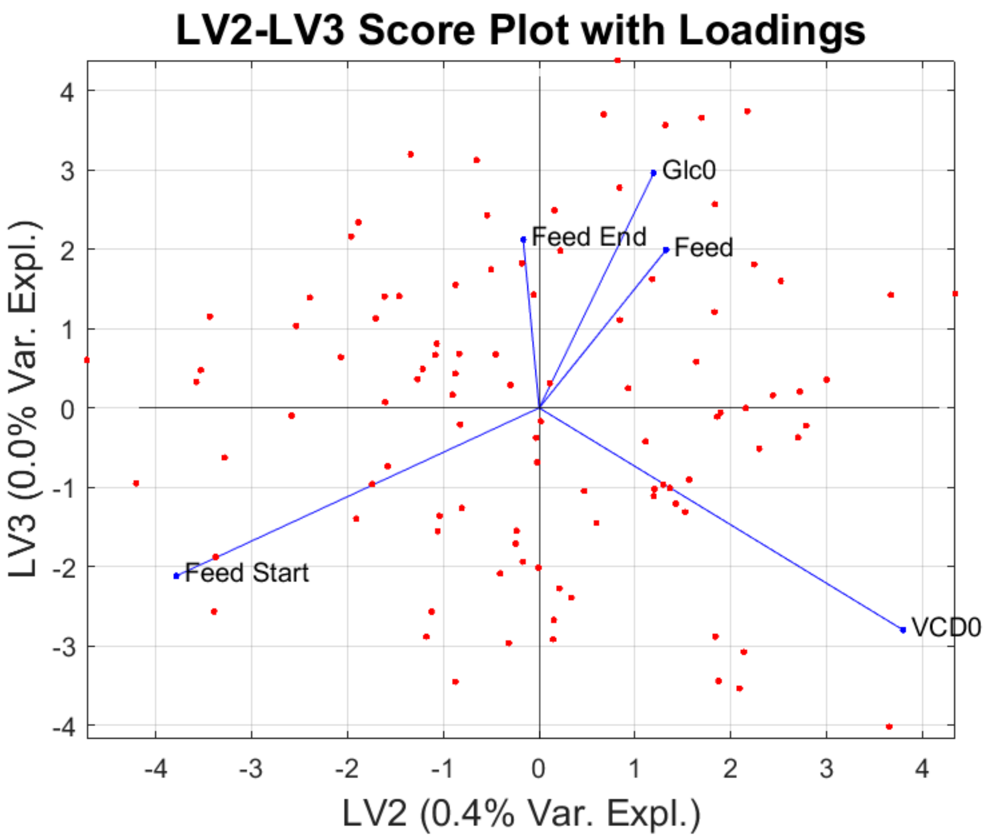
```
N_LV_plot = min(N_LV,3);
x_label_str = cell(N_LV_plot,1);
for nlv_1st = 1:N_LV_plot
    x_label_str{nlv_1st} = ['LV',num2str(nlv_1st),' (',num2str(100*pct_var(2,nlv_1st),'%4.1f'),
end
pred_var_names = {'Glc0','Feed Start','Feed End','Feed','VCD0'};
for nlv_1st = 1:N_LV_plot-1
    for nlv_2nd = nlv_1st+1:N_LV_plot
        figure, clf
        v = [nlv_1st,nlv_2nd];
        biplot(x_loadings(:,v),'scores',x_scores(:,v),'varlabels',pred_var_names);
        box on
        title(['LV',num2str(nlv_1st),'-LV',num2str(nlv_2nd),' Score Plot with Loadings'],'FontS
        xlabel(x_label_str{nlv_1st},'FontSize',14), ylabel(x_label_str{nlv_2nd},'FontSize',14)
    end
end
```



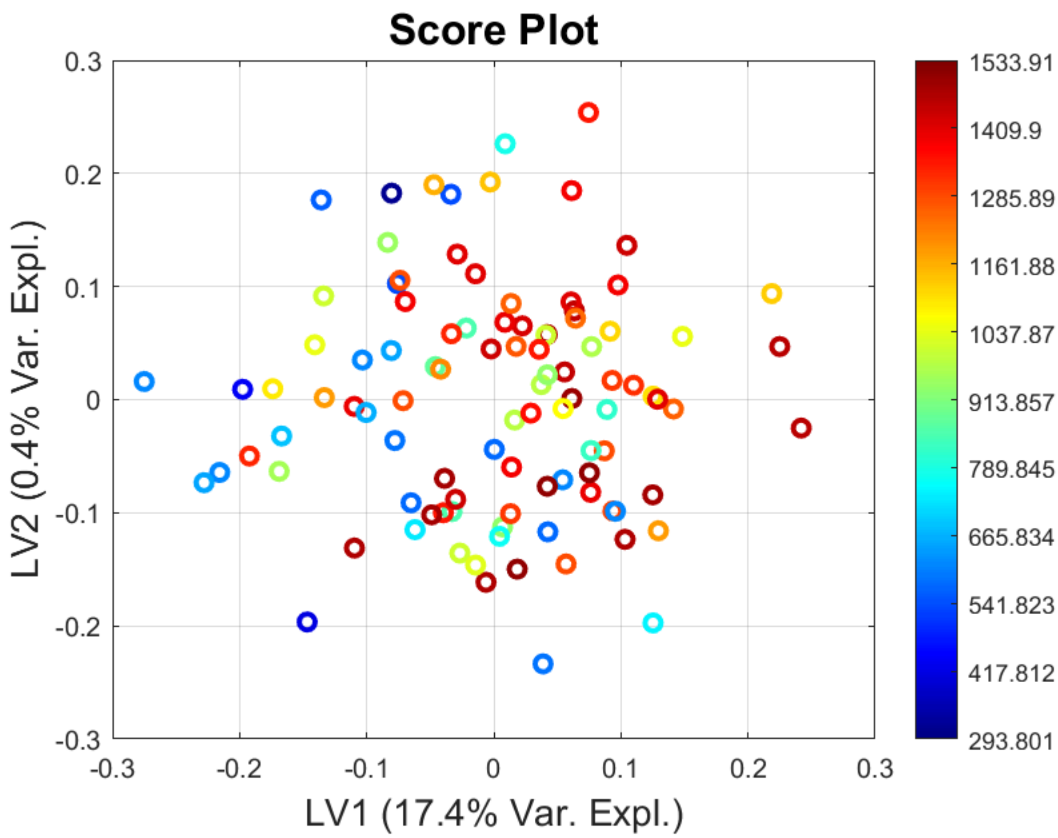


Plot scores based on final titer

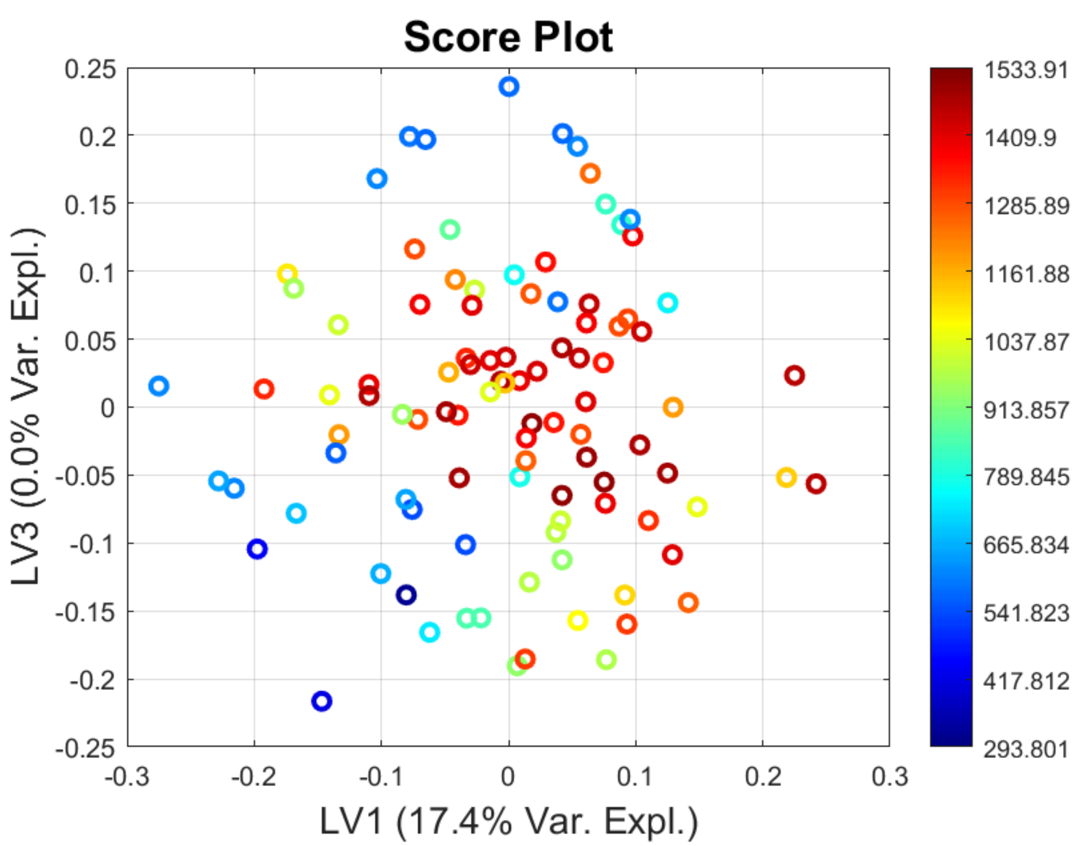
```
color_map = jet(101); colormap(jet)
```



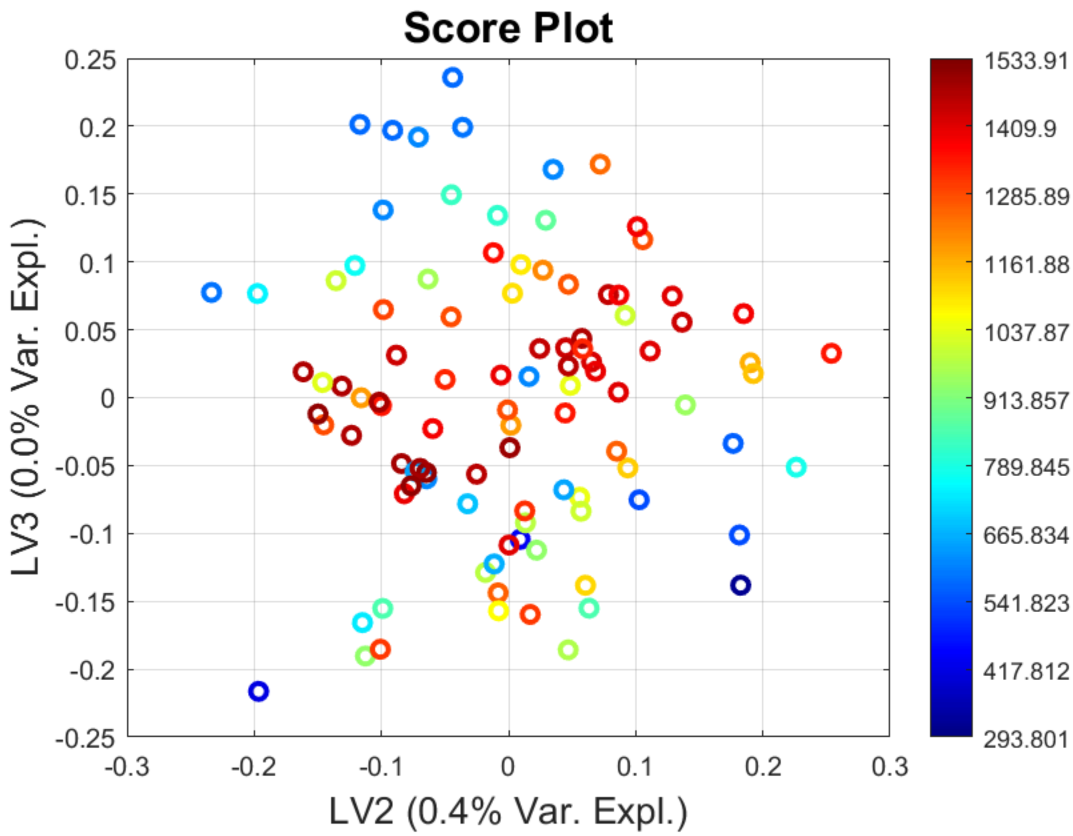

```
line_color = zeros(Nruns,3);
for nexp = 1:Nruns
    v = 1+round(100*(f_DoE(nexp)-min(f_DoE))/(max(f_DoE)-min(f_DoE)));
    line_color(nexp,:) = color_map(v,:);
end
for nlv_1st = 1:N_LV_plot-1
    for nlv_2nd = nlv_1st+1:N_LV_plot
        figure, clf, hold on
        scatter(x_scores(:,nlv_1st),x_scores(:,nlv_2nd),'o',"CData",line_color,'LineWidth',2)
        box on, grid on
        xlabel(x_label_str{nlv_1st},'FontSize',14), ylabel(x_label_str{nlv_2nd},'FontSize',14)
        title('Score Plot','FontSize',16)
        colormap(jet),h = colorbar; h.TickLabels = num2cell(linspace(min(f_DoE),max(f_DoE),11))
    end
end
```



Score Plot
LV2 (0.4% Var. Expl.)
LV1 (17.4% Var. Expl.)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.3 -0.2 -0.1 0 0.1 0.2 0.3
1533.91
1409.9
1285.89
1161.88
1037.87
913.857
789.845
665.834
541.823
417.812
293.801

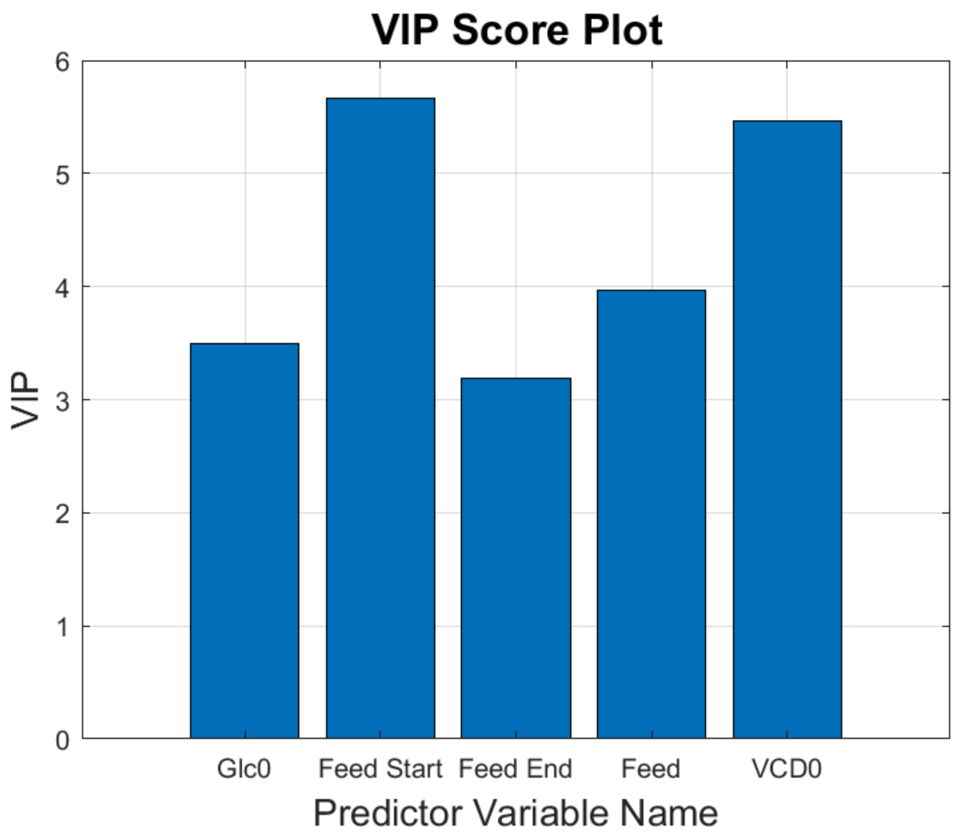


Score Plot
LV3 (0.0% Var. Expl.)
LV1 (17.4% Var. Expl.)
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3 -0.2 -0.1 0 0.1 0.2 0.3
1533.91
1409.9
1285.89
1161.88
1037.87
913.857
789.845
665.834
541.823
417.812
293.801



Plot VIP scores

```
W0 = stats.W ./ sqrt(sum(stats.W.^2,1));
sumSq = sum(x_scores.^2,1).*sum(x_loadings.^2,1);
VIP = sqrt(Nruns*sum(sumSq.*(W0.^2),2)./sum(sumSq,2));
figure, clf
h = bar(VIP);
box on, grid on
xlabel('Predictor Variable Name','FontSize',14),ylabel('VIP','FontSize',14)
title('VIP Score Plot','FontSize',16)
h.Parent.XTickLabel = pred_var_names;
```



Simulate Cross-Validation

In this section, we will simulate a typical cross- validation to define the optimal number of latent variables.

Define the number of folds

First define the number of folds.

```
Nfolds = 5;
fold_no = mod(0:Nruns-1,Nfolds)+1
```

```
fold_no = 1×100
     1     2     3     4     5     1     2     3     4     5     1     2     3 ···
```

Compute cross-validation

Nfold PLS1 models are trained using (Nfolds-1) folds. For each model, the sum of squared residuals (SSR) is calculated and summed up.

This is repeated for different numbers of latent variables. The number of latent variables returning the least value of the SSRs is chosen as optimal.

A second criterion is selected, namely the Bayesian Information Criterion (BIC), which is weighting the effect of the number of latent variables, i.e., if two values of the number of latent variables are returning a similar value of the SSR, then the one using less variables is chosen to be more likely to produce robust predictions.

```
RMSE_CV = zeros(5,1);
```

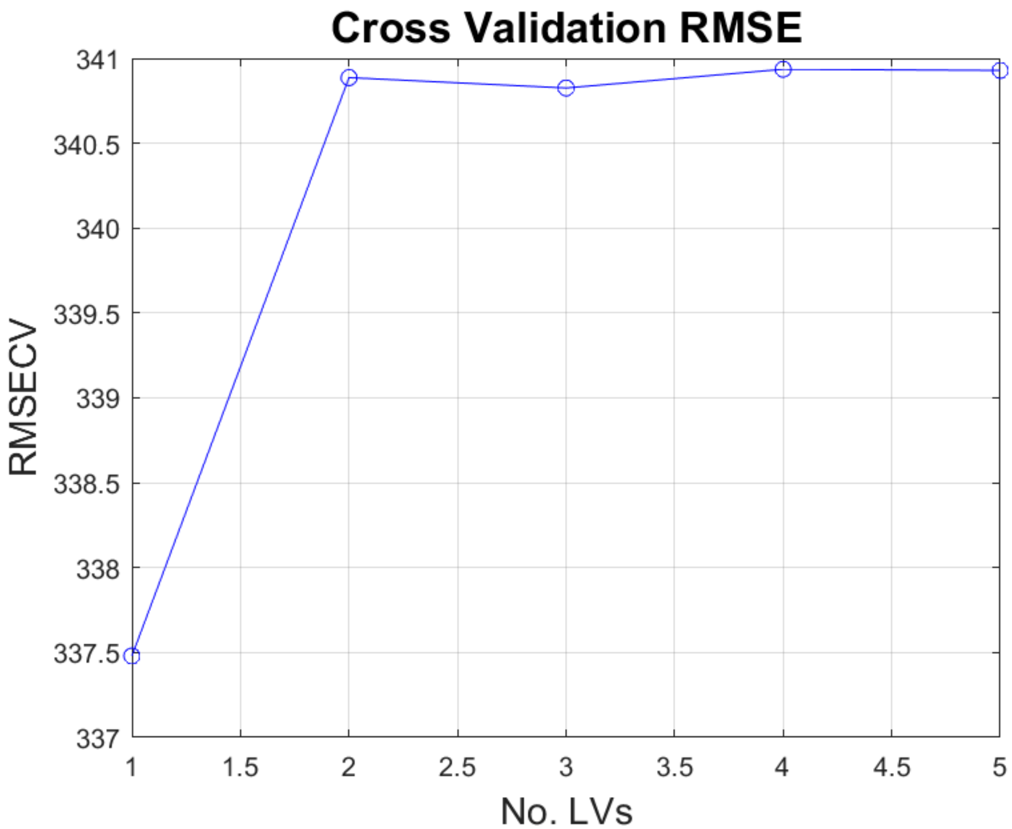
```
BIC_CV = zeros(5,1);
for nlv = 1:5
    SSE = 0;
    for nf = 1:Nfolds
        % get training data
        v = fold_no ~= nf;
        Xs = DoE_nondim(v,:);
        y = f_DoE(v);
        % train PLS
        [~,~,~,~,beta] = plsregress(Xs,y,nlv);
        % predict validation set
        v = ~v;
        Xs = DoE_nondim(v,:);
        y = f_DoE(v);
        y_fit = [ones(nnz(v),1),Xs]*beta;
        % compute RMSE_CV
        SSE = SSE+sum((y-y_fit).^2);
    end
    RMSE_CV(nlv) = sqrt(SSE/Nruns);
    BIC_CV(nlv) = Nruns*log(SSE/Nruns)+nlv*log(Nruns);
end
```

Plot results

```
RMSE_CV
```

```
RMSE_CV = 5×1
  337.4796
  340.8866
  340.8255
  340.9358
  340.9292
```

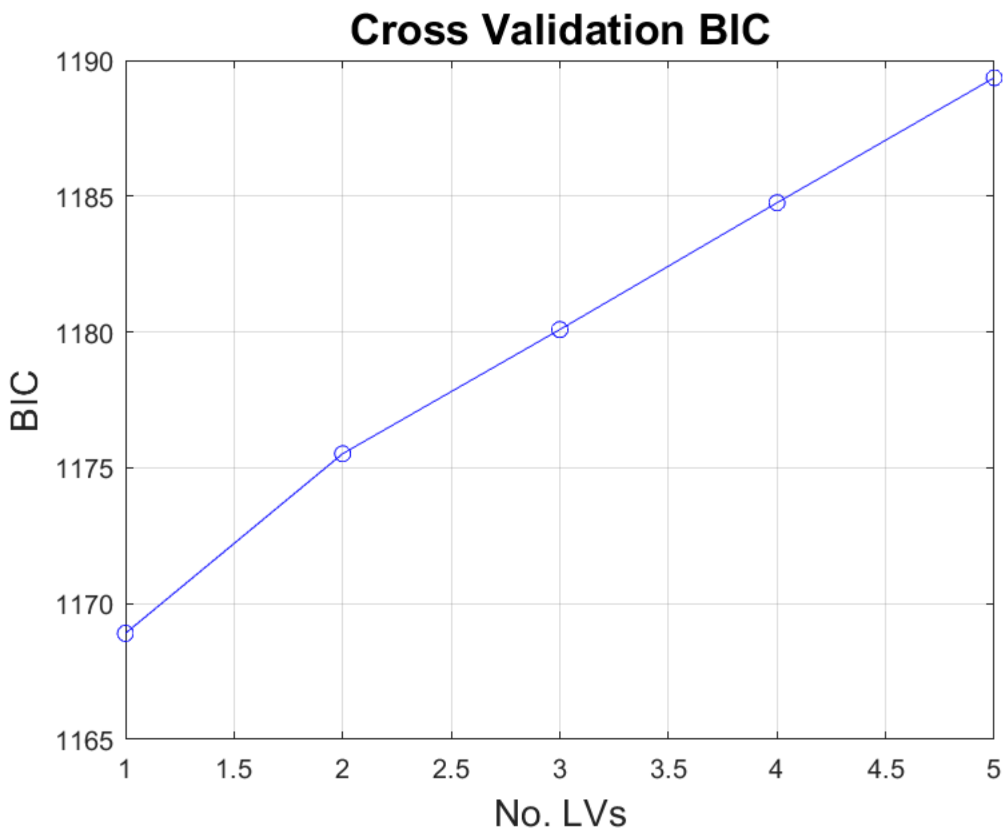
```
figure, clf
plot(1:5,RMSE_CV,'b-o');
box on, grid on
xlabel('No. LVs','FontSize',14),ylabel('RMSECV','FontSize',14)
title('Cross Validation RMSE','FontSize',16)
```




```
BIC_CV
```

```
BIC_CV = 5×1
10^3 ×
    1.1689
    1.1755
    1.1801
    1.1848
    1.1894
```

```
figure, clf
plot(1:5,BIC_CV,'b-o');
box on, grid on
xlabel('No. LVs','FontSize',14),ylabel('BIC','FontSize',14)
title('Cross Validation BIC','FontSize',16)
```



Historical PLS Models

In an so-called historical model, the data from different experiments are ordered into a batch-wise unfolded (BWU) matrix (i.e., every row corresponds to an experiment).

The BWU can be used to compute final properties of the experiment, like CQAs, which are typically the effect of the cumulated effect of the experiment profile.

In this example, we will use the BWU matrix to predict the final value of titer. Clearly, titer information are removed from the BWU matrix.

Define the number of days to include

Here the use can select the amount of days to include in the historical model.

```
Ndays = 13;
```

Define the number of latent variables

```
N_LV  = 5;
```

Create the BWU matrix

Here the BWU matrix is created. The vaues of the manipulated variables are added as columns at the beginning of the matrix.

```
BWU_hist = [DoE_LHD(:,2:3),BWU];
day_no = [0,0];
var_name = {'Feed Start','Feed End'};
active_vars = 1:77;
for n = 0:14
    v = 3+5*n:2+5*(n+1);
    day_no(1,v) = n;
    var_name(1,v) = {['VCD@d',num2str(n)],['Glc@d',num2str(n)],['Lac@d',num2str(n)],['Titer@d',
end
```

Remove titer

```
v = 6:5:76;
BWU_hist(:,v) = [];
day_no(:,v) = [];
var_name(:,v) = [];
active_vars(:,v) = [];
```

Remove exceeding days

```
v = day_no <= Ndays;
day_no = day_no(v);
var_name = var_name(v);
BWU_hist = BWU_hist(:,v)
```

```
BWU_hist = 100×58
    2.0000   12.0000    0.6534   24.9894         0         0    1.9990   23.8281 ···
    4.0000   12.0000    0.7677   34.4164         0         0    2.3632   33.0446
    3.0000    9.0000    0.8983   64.0989         0         0    2.7817   62.4839
    2.0000    9.0000    0.3519   15.4365         0         0    1.0650   14.8178
    3.0000    9.0000    0.5413   15.7769         0         0    1.6240   14.8281
    2.0000    9.0000    0.4832   34.9041         0         0    1.5108   34.0351
    3.0000    8.0000    0.9639   30.5586         0         0    2.9255   28.8476
    3.0000   11.0000    0.7033   13.9402         0         0    2.0731   12.7188
    3.0000   11.0000    0.5150   68.3358         0         0    1.6298   67.4016
    2.0000   10.0000    0.8565   20.1392         0         0    2.5655   18.6336
      ⋮
```

```
active_vars = active_vars(v);
```

Eliminate invariant columns

```
v = (std(BWU_hist) > 1e-8) & (std(BWU_hist)./mean(BWU_hist) > 1e-4);
day_no = day_no(v);
var_name = var_name(v);
BWU_hist = BWU_hist(:,v);
active_vars = active_vars(v);
```

Remove linearly dependent columns

```
lic = find_linearly_independent_columns(BWU_hist);
day_no = day_no(lic);
var_name = var_name(lic);
BWU_final = BWU_hist(:,lic);
```

```
active_vars = active_vars(lic);
```

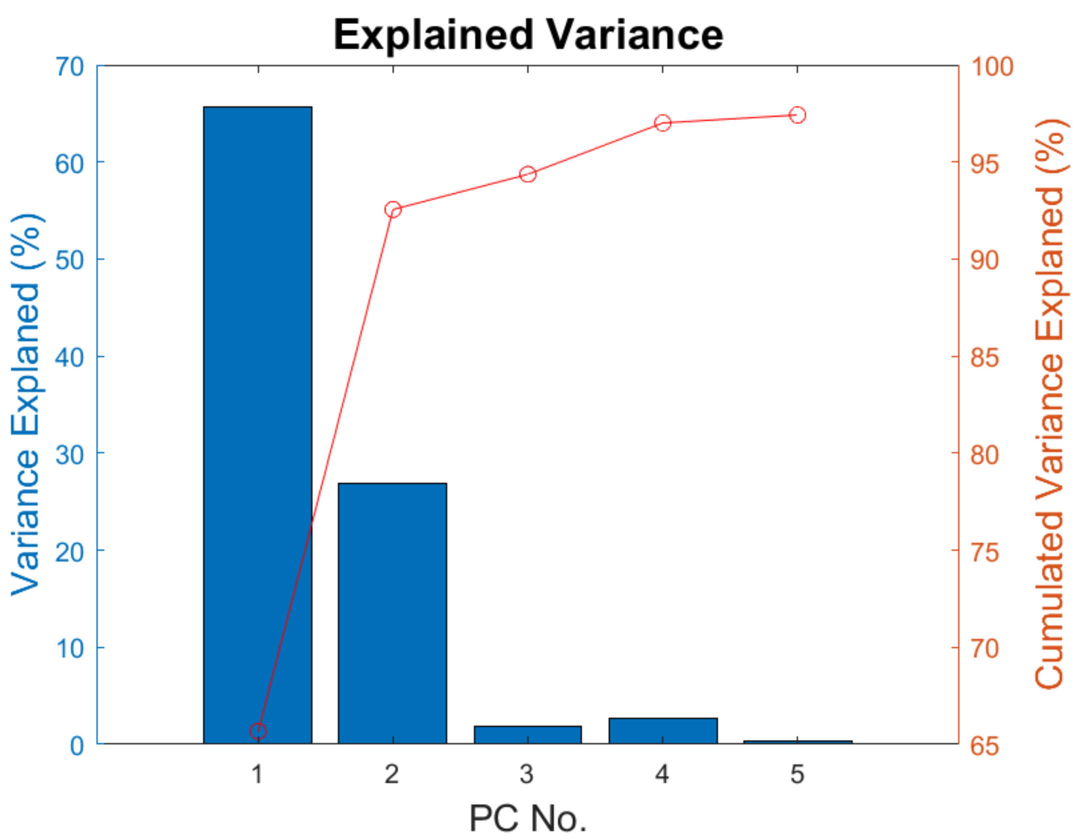
Create model

Create a PLS model from the initial design to the final titer

```
BWU_mean = mean(BWU_final);
BWU_std = std(BWU_final);
Xs = (BWU_final-repmat(BWU_mean,size(BWU_final,1),1))./repmat(BWU_std,size(BWU_final,1),1);
[x_loadings,y_loadings,x_scores,y_scores,beta,pct_var,~,stats] = plsregress(Xs,f_DoE,N_LV);
```

Plot results

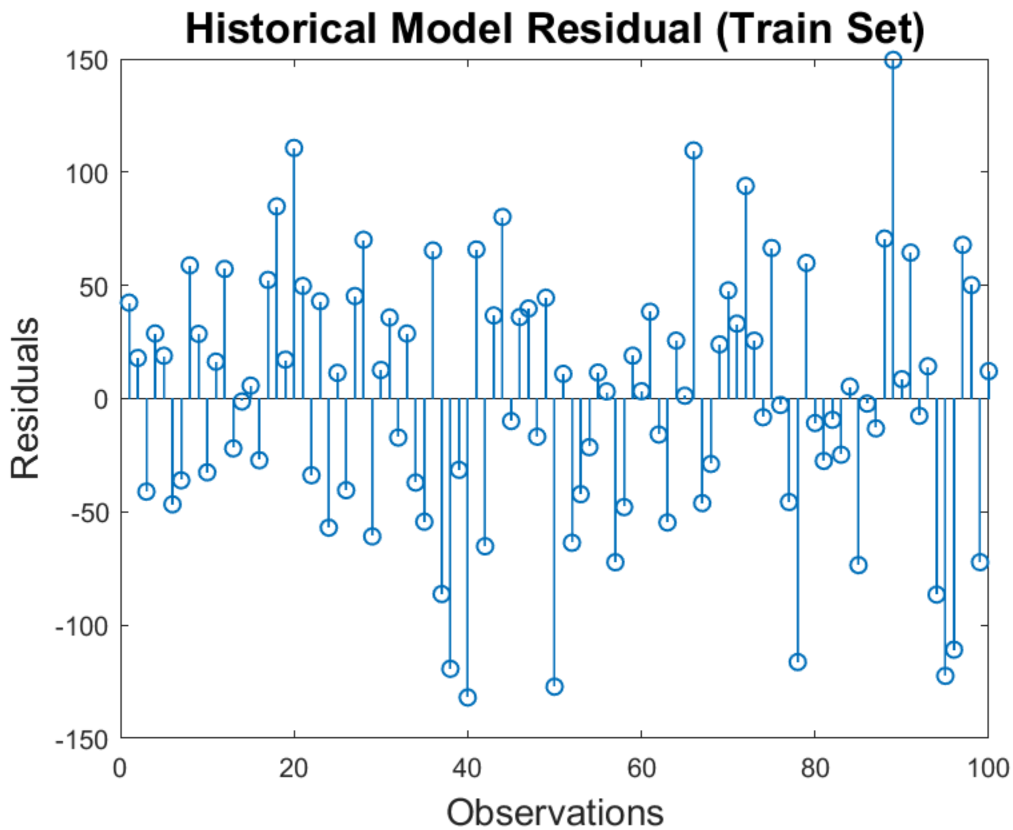
```
figure, clf
yyaxis left
bar(1:N_LV,100*pct_var(2,:))
xlabel('PC No.','FontSize',14),ylabel('Variance Explaned (%)','FontSize',14)
title('Explained Variance','FontSize',16)
yyaxis right
plot(1:N_LV,100*cumsum(pct_var(2,:)),'ro-')
ylabel('Cumulated Variance Explaned (%)','FontSize',14)
```



Compute fitted response residuals

```
yfit = [ones(Nruns,1),Xs]*beta;
residuals = f_DoE - yfit;
figure, clf
stem(residuals,'LineWidth',1)
```

```
xlabel('Observations','FontSize',14), ylabel('Residuals','FontSize',14);
title('Historical Model Residual (Train Set)','FontSize',16)
```




```
SSE = sum(residuals.^2);
RMSE_abs = sqrt(SSE/Nruns)
```

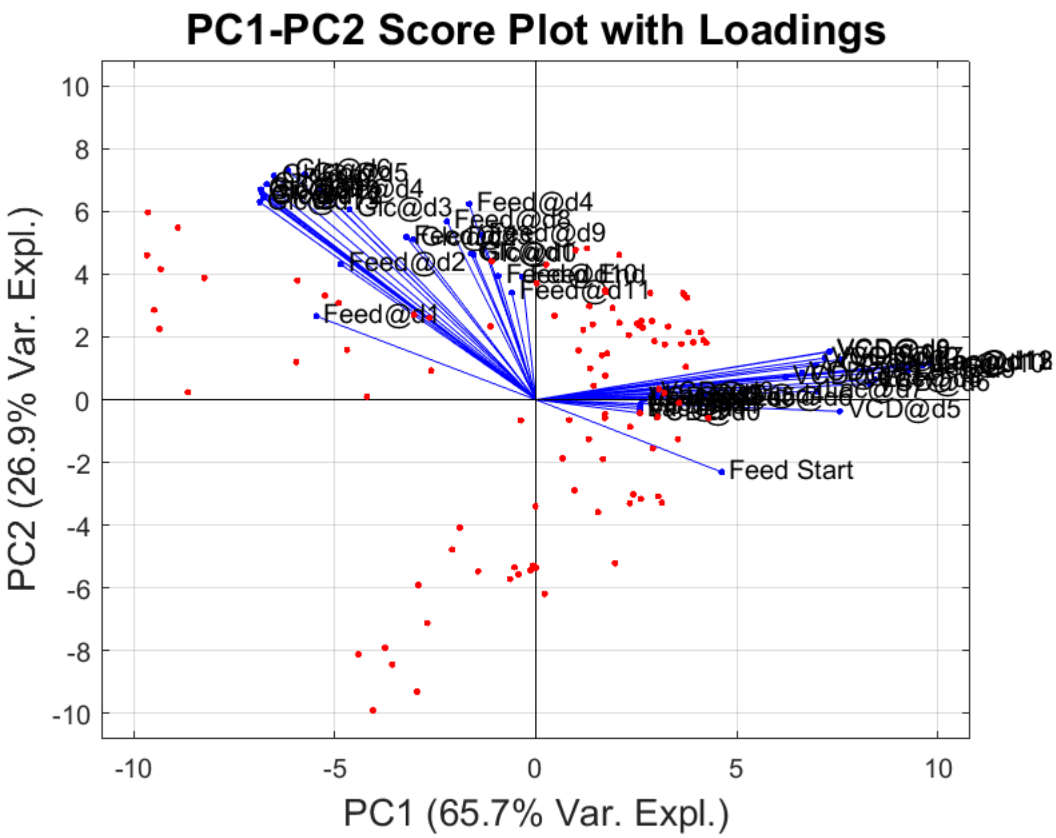
```
RMSE_abs = 55.6298
```

```
RMSE_rel = RMSE_abs/std(f_DoE_test)
```

```
RMSE_rel = 0.1660
```

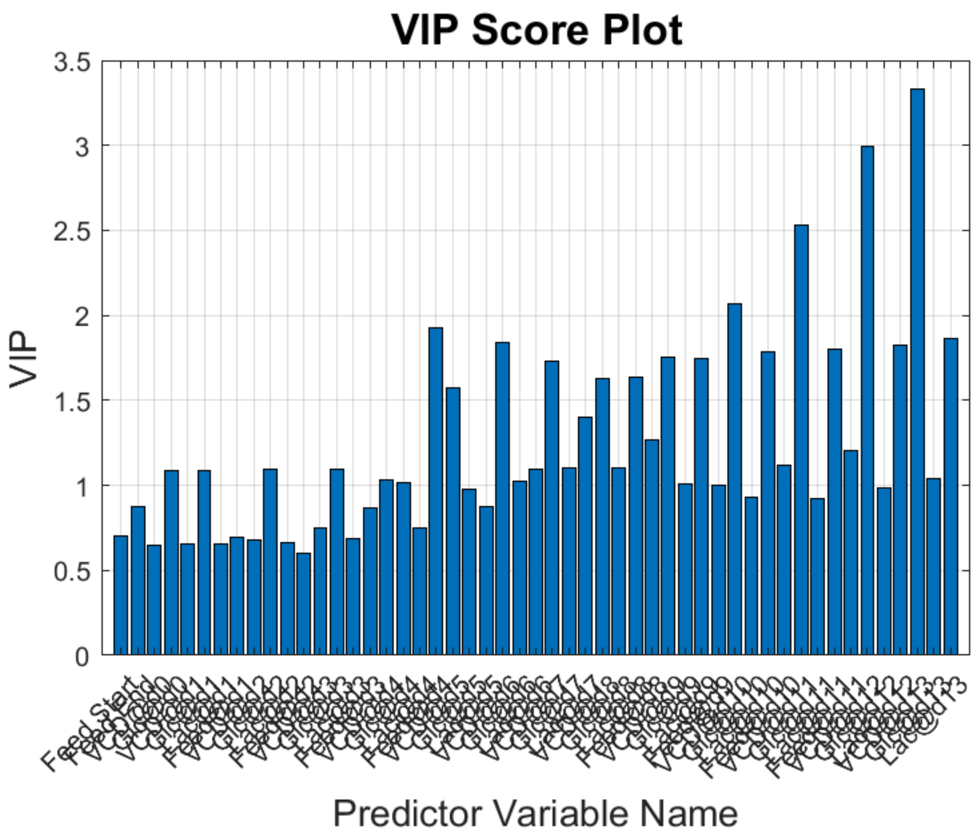
Plot scores (normalized) and loadings for the predictors

```
figure, clf
biplot(x_loadings(:,1:2),'scores',x_scores(:,1:2),'varlabels',var_name);
box on
xlabel(['PC1 (',num2str(100*pct_var(2,1),'%4.1f'),'% Var. Expl.)'],'FontSize',14)
ylabel(['PC2 (',num2str(100*pct_var(2,2),'%4.1f'),'% Var. Expl.)'],'FontSize',14)
title('PC1-PC2 Score Plot with Loadings','FontSize',16)
```



Plot VIP scores

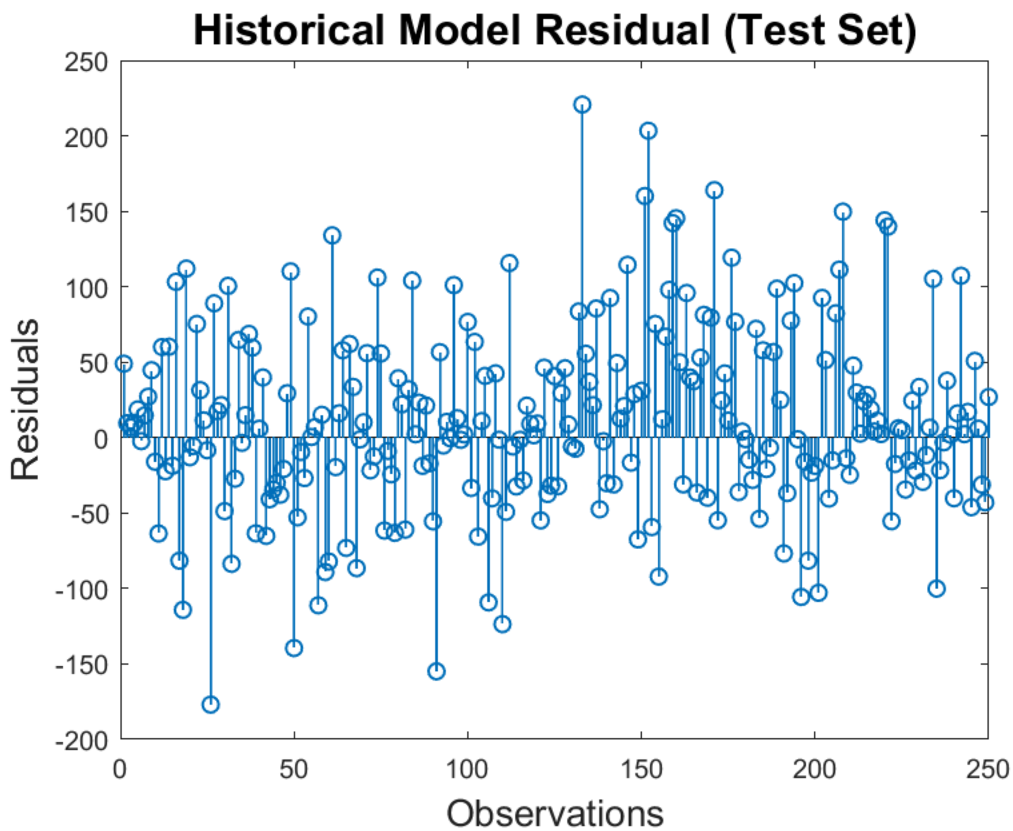
```
W0 = stats.W ./ sqrt(sum(stats.W.^2,1));
sumSq = sum(x_scores.^2,1).*sum(x_loadings.^2,1);
VIP = sqrt(Nruns*sum(sumSq.*(W0.^2),2)./sum(sumSq,2));
figure, clf
h = bar(VIP);
box on, grid on
xlabel('Predictor Variable Name','FontSize',14),ylabel('VIP','FontSize',14)
title('VIP Score Plot','FontSize',16)
h.Parent.XTick = 1:size(Xs,2);h.Parent.XTickLabel = var_name;h.Parent.XTickLabelRotation = 45;
```



Test model

The historical model created above is tested on the test set.

```
BWU_hist = [DoE_LHD_test(:,2:3),BWU_test];
BWU_final = BWU_hist(:,active_vars);
Xs = (BWU_final-repmat(BWU_mean,size(BWU_final,1),1))./repmat(BWU_std,size(BWU_final,1),1);
yfit = [ones(Nruns_test,1),Xs]*beta;
residuals = f_DoE_test - yfit;
figure, clf
stem(residuals,'LineWidth',1)
xlabel('Observations','FontSize',14), ylabel('Residuals','FontSize',14);
title('Historical Model Residual (Test Set)','FontSize',16)
```




```
SSE = sum(residuals.^2);
RMSE_abs = sqrt(SSE/Nruns_test)
```

```
RMSE_abs = 63.1760
```

```
RMSE_rel = RMSE_abs/std(f_DoE_test)
```

```
RMSE_rel = 0.1885
```

```
function lic = find_linearly_independent_columns(BWU)

X = zscore(double(BWU));
[~,R,E] = qr(X,0);
diagr = abs(diag(R));
tol = 1e-8;
r = find(diagr >= tol*diagr(1),1,'last');
lic = sort(E(1:r));

end
```