Comparison of two second degree designs

(c) jean-marie.furbringer@epfl.ch sept 2021

Table of Contents

The reference model

Setup of the reference model

A reference model is chosen to simulate the experiments. It is a quadratic model at 3 dimensions

$$y = a_o + a_1x_1 + a_2x_2 + a_3x_3 + a_{12}x_1x_2 + a_{13}x_1x_3 + a_{23}x_2x_3 + a_{11}x_1^2 + a_{22}x_2^2 + a_{33}x_3^2$$

Coefficients of the reference model are chosen at random. This allows a honest comparison between the designs.

```
%kappa=[.4 -.5 .3 -.2 .7 -.6 -.5 .8 -.9 -.75];
 kappa=rand(1,10);
```

An experimental variance is used for the simulation of the experiment:

```
sigma=0.05;
disp(['sigma = ',num2str(sigma)])
```

```
sigma = 0.05
```

Visualization of the reference model

To visualize the reference model that we attemp to fit, let's draw a slice view and an isovalue representation. To do that it is necessary to

1) declare the symbolic variables x_1, x_2, x_3,

```
syms x_1 x_2 x_3
assume(x_1,'real')
assume(x_2,'real')
assume(x_3,'real')
```

2) build the model coordinate as symbolic vector ,

$$f(x) = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ x_1x_2 \\ x_1x_3 \\ x_2x_3 \\ x_1^2 \\ x_2^2 \\ x_3^2 \end{pmatrix}$$

```
f=[1;x_1;x_2;x_3;x_1*x_2;x_1*x_3;x_2*x_3;x_1^2;x_2^2;x_3^2];
```

3) build the model as the product of the coefficients and the model coordinate, $y = \alpha \cdot f(x)$, ce qui donne:

```
y=kappa*f
```

```
y =
```

$$\frac{153933462881711\,x_1^2}{281474976710656} + \frac{1423946432832521\,x_1\,x_2}{2251799813685248} + \frac{109820732902227\,x_1\,x_3}{1125899906842624} + \frac{8158648460577917\,x_1}{9007199254740992} + \frac{8(}{}$$

4) define the domain,

```
% domain
min_x=-1;
max_x=1;
```

5) define the number of points per interval

```
% number of points per dimension
N=21;
disp(['N = ',num2str(N)])
```

```
N = 21
```

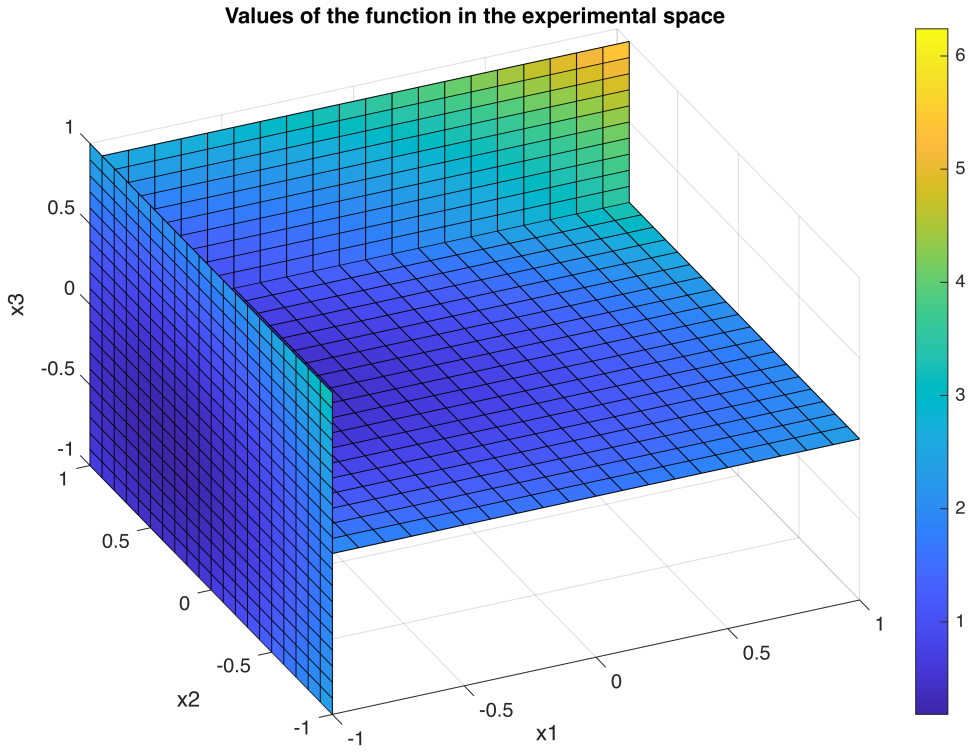
6) compute the values of the model on a grid defined with 3 3D-matrices X_1, X_2 and X_3 (one per dimension),

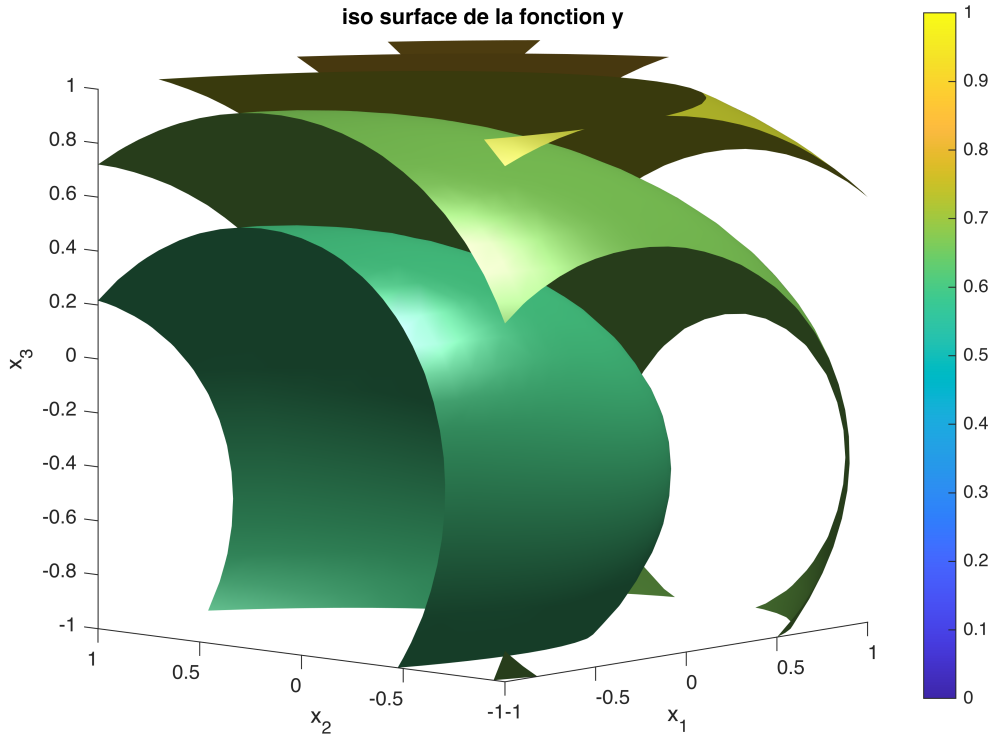
```
% factor values
[X1,X2,X3]=meshgrid(linspace(min_x,max_x,N));

% position of the planes for the slices

p1=-1;
p2=0.9;
p3=0;
```

```
Xs=[p1 p2 p3];

% evaluation of the model
Y_ref=double(subs(y,{x_1,x_2,x_3},{X1,X2,X3}));
```

7) visualize the model with the function viz*() defined at the end of this file*

```
viz(X1,X2,X3,Y_ref,Xs)
```





This ends the setup of the experimental "reality". Obviously, in the real situation, those informations are not known by the experimenter.

Now let's start with the experimenter mimicking.

Strategy 1: composite design

Let's built a *factorial design* for 3 factors:

```
E1=(ff2n(3)-.5)*2;
disp(array2table(E1,"VariableNames",{'X_1','X_2','X_3'}))
```

X_1	X_2	X_3
-1	-1	-1
-1	-1	1
-1	1	-1
-1	1	1
1	-1	-1
1	-1	1
1	1	-1
1	1	1

Let's built a *star design* for 3 factors:

```
star= [1 0 0
      -1 0 0
       0 1 0
```

```
        0 -1 0
        0 0 1
        0 0 -1];
```

Ratio of the distance of the star points in relation with the factorial points

```
alfa=1.215;
E2=alfa*star;
disp(array2table(E2,"VariableNames",{'X_1','X_2','X_3'}))
```

X_1	X_2	X_3
1.215	0	0
-1.215	0	0
0	1.215	0
0	-1.215	0
0	0	1.215
0	0	-1.215

Set up the central composite design with one central experiment

```
E= [E1;E2;0 0 0];
disp(array2table(E,"VariableNames",{'X_1','X_2','X_3'}))
```

X_1	X_2	X_3
-1	-1	-1
-1	-1	1
-1	1	-1
-1	1	1
1	-1	-1
1	-1	1
1	1	-1
1	1	1
1.215	0	0
-1.215	0	0
0	1.215	0
0	-1.215	0
0	0	1.215
0	0	-1.215
0	0	0

Set up of the model matrix

Model matrix

```
M=x2fx(E,'quadratic');
```

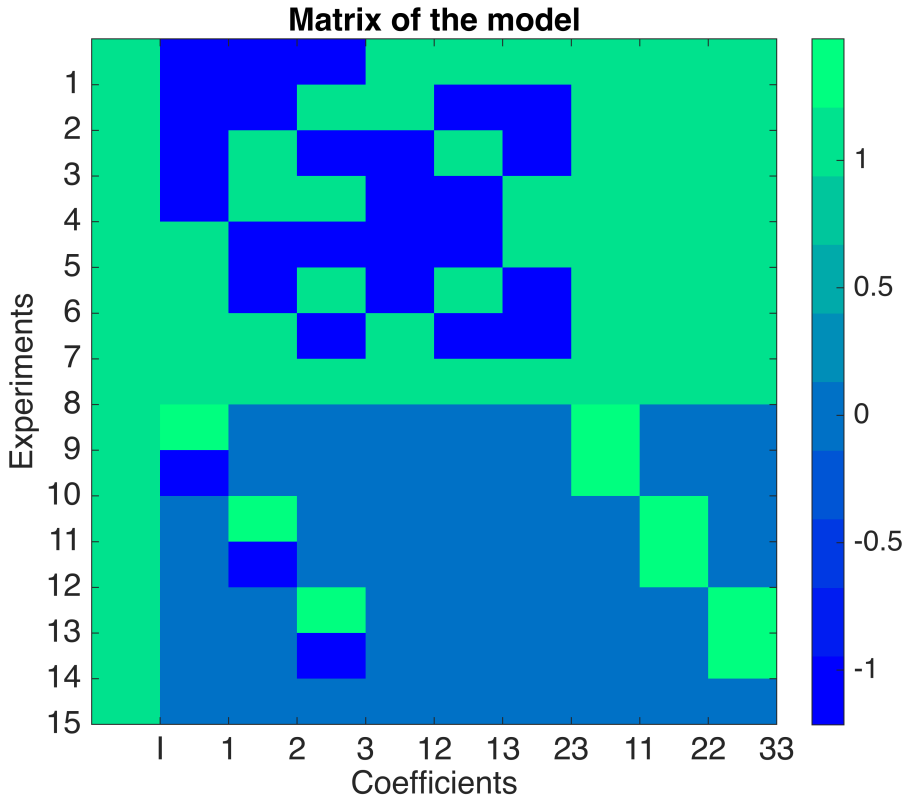
Dispertion matrix $D = (X'X)^{-1}$

```
D=inv(M'*M);
```

Representation of the model matrix

```
figure
```

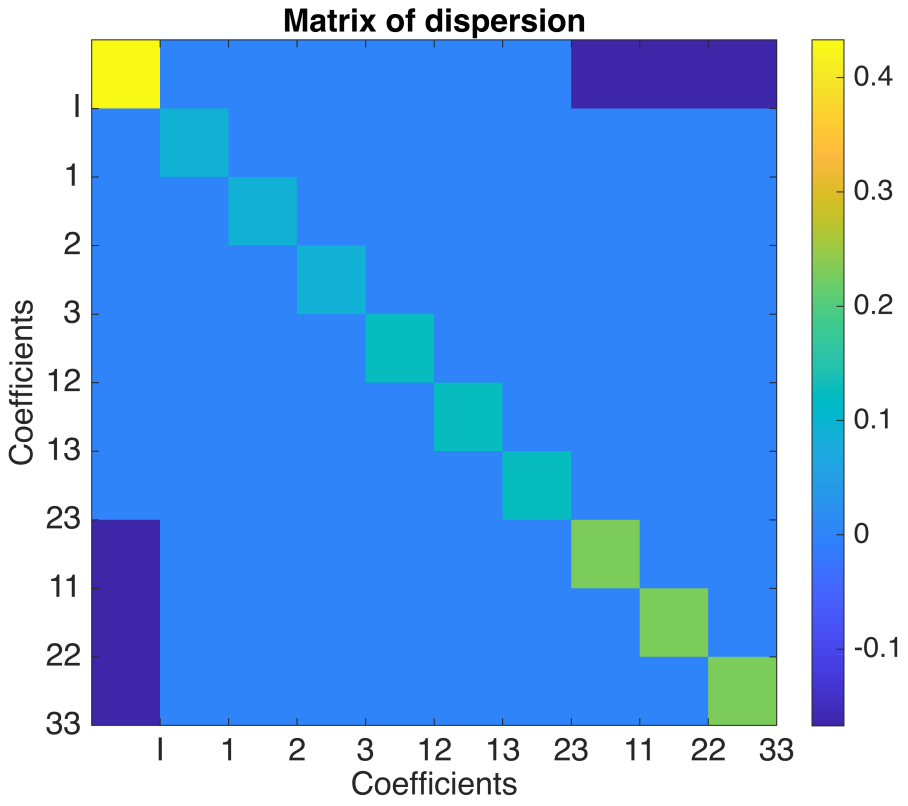
```
imagesc(M),colorbar
set(gca, 'Xtick',.5:1:10.5,...
'Ytick',.5:1:15.5,...
'FontSize',16,...
'XTickLabel',{' ' 'I' '1' '2' '3' '12' '13' '23' '11' '22' '33'},...
'YTickLabel',{ ' ' '1' '2' '3' '4' '5' '6' '7' '8' '9' '10' '11' '12' '13'
'14' '15'})
colormap(winter(10))
axis ij
axis square
title('Matrix of the model','FontSize',16)
xlabel('Coefficients','FontSize',16)
ylabel('Experiments','FontSize',16)
```



Representation of the dispertion matrix

```
figure
imagesc(D),colorbar
set(gca, 'Xtick',.5:1:10.5,...
'Ytick',.5:1:10.5,...
'FontSize',16,...
'XTickLabel',{' ' 'I' '1' '2' '3' '12' '13' '23' '11' '22' '33'},...
'YTickLabel',{ ' ' 'I' '1' '2' '3' '12' '13' '23' '11' '22' '33'})
%c with olormap(pink(10))
axis ij
axis square
title('Matrix of dispersion','FontSize',16)
```

```
xlabel('Coefficients','FontSize',16)
ylabel('Coefficients','FontSize',16)
```



The structure of the dispersion matrix is typical from a quadratic model orthogonalyty for the first main effects and the interactions and the correlation with the constant for the quadratic terms. The computing of the alias matrix between the linear model with interactions and the quadratic terms shows that if the quadratic terms are not evaluated their values will biased the constant with 73 percent of the quadratic curvature.

```
M1=M(:,1:7);
M2=M(:,8:10);
M_Alias=(M1'*M1)\(M1'*M2);
disp(M_Alias)
```

```
    0.7302    0.7302    0.7302
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
```

Measurement simulation

The measurement is simulated by adding a random error to the real model

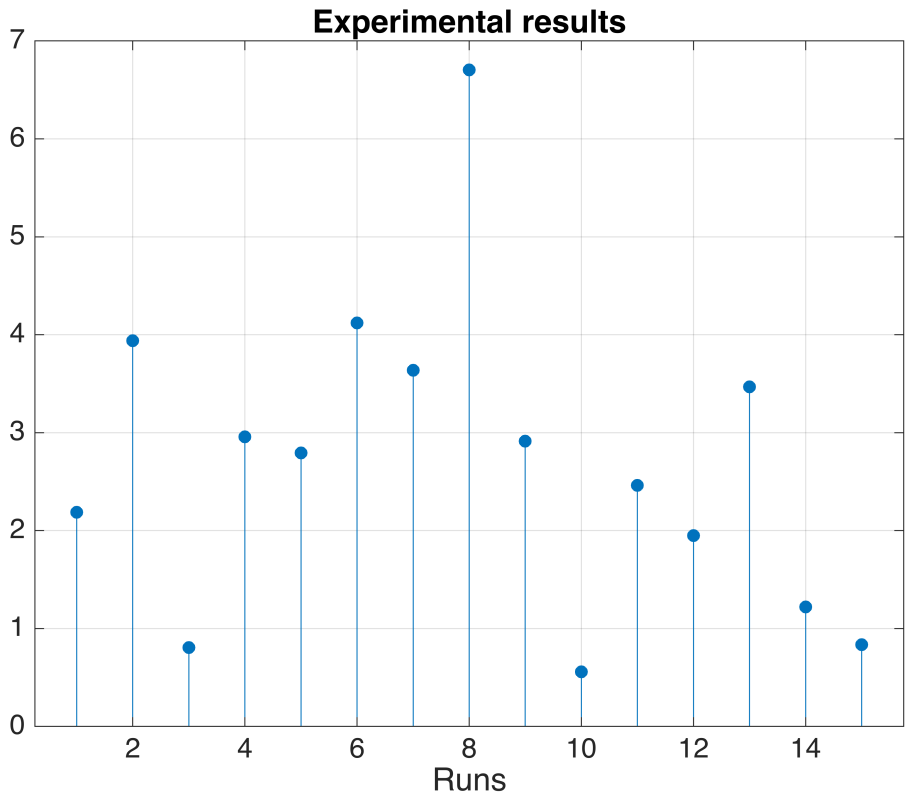
$$Y = X\kappa + \epsilon_i$$

```
Nexp=size(M,1); % number of experiments
```

```
Y = M*kappa'.*(ones(Nexp,1)+randn(Nexp,1)*sigma);
```

Graphical representation of the experimental results

```
figure
stem(Y,"filled")
set(gca,'FontSize',14)
title('Experimental results','FontSize',16)
xlabel('Runs','FontSize',16)
grid on
```



Computing of the coefficients

The coefficients are computed by a least square fit algorithm

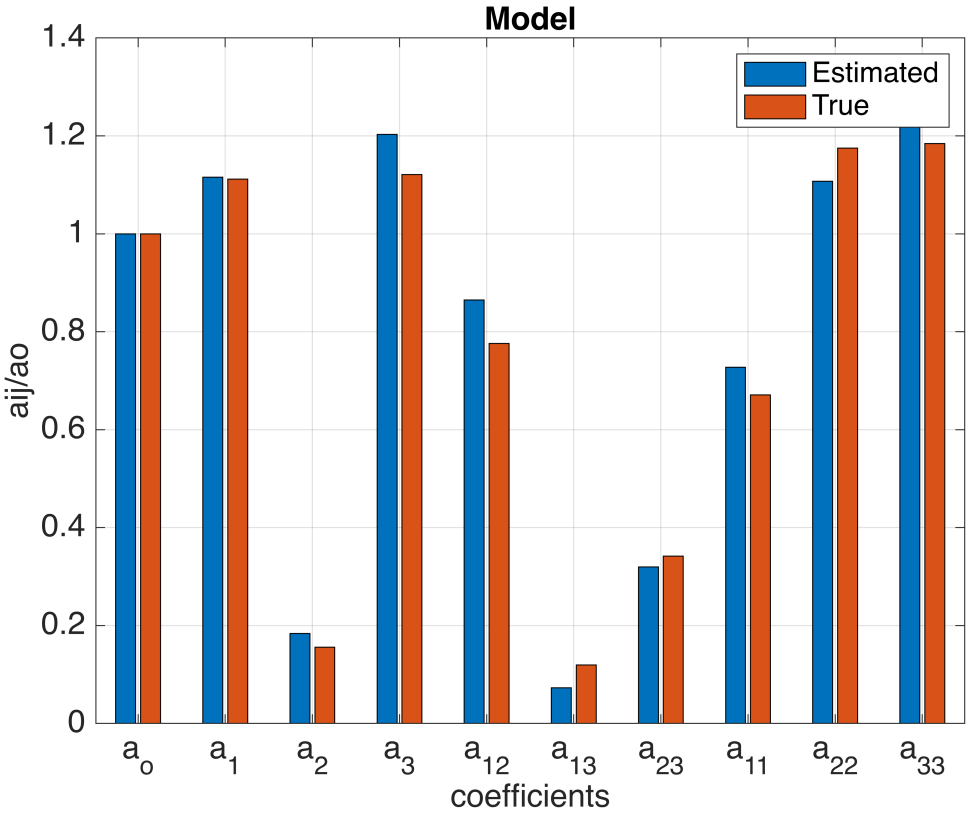
$\hat{\beta}=(X'X)^{-1} X' Y$

```
beta=D*M'*Y;
```

and then they can be compared with the original ones

```
figure
bar([beta/beta(1), kappa'/kappa(1)])
set(gca,'FontSize',16,...
    'XTickLabel',{'a_o' 'a_1' 'a_2' 'a_3' ...
    'a_{12}' 'a_{13}' 'a_{23}' 'a_{11}' 'a_{22}' 'a_{33}'})
title('Model','FontSize',16)
```

```
xlabel('coefficients','FontSize',16)
ylabel('aij/ao','FontSize',16)
legend("Estimated", "True")
grid on
```




```
mdl_1=fitlm(E,Y,"quadratic")
```

```
mdl_1 =
Linear regression model:
    y ~ 1 + x1*x2 + x1*x3 + x2*x3 + x1^2 + x2^2 + x3^2

Estimated Coefficients:
                   Estimate       SE        tStat       pValue
                   ________    ________    _______    __________

    (Intercept)     0.83686     0.15306     5.4676     0.0027864
    x1              0.93357    0.070293     13.281    4.3261e-05
    x2              0.15398    0.070293     2.1906      0.080039
    x3               1.0067    0.070293     14.322    2.9916e-05
    x1:x2           0.72362    0.082247     8.7981    0.00031481
    x1:x3          0.061095    0.082247    0.74282         0.491
    x2:x3           0.26743    0.082247     3.2515      0.022659
    x1^2            0.60887     0.11139      5.466     0.0027898
    x2^2             0.9267     0.11139     8.3193     0.0004101
    x3^2             1.0213     0.11139     9.1681    0.00025887


Number of observations: 15, Error degrees of freedom: 5
Root Mean Squared Error: 0.233
R-squared: 0.992,  Adjusted R-Squared: 0.979
F-statistic vs. constant model: 73.1, p-value = 9.02e-05
```

```
anova(mdl_1,"summary",2)
```

ans = 5×5 table

	SumSq	DF	MeanSq	F	pValue
1 Total	35.8904	14	2.5636	NaN	NaN
2 Model	35.6198	9	3.9578	73.1337	9.0202e-05
3 . Linear	20.9053	3	6.9684	128.7665	3.7427e-05
4 . Nonlinear	14.7145	6	2.4524	45.3173	3.3619e-04
5 Residual	0.2706	5	0.0541	NaN	NaN

Canonical analysis

The first step consists in expressing the model in terms of constant a_o, linear vector $\vec{a}$ and quadratic matrix A:

```
% Routine belonging to LISA library
ao=beta(1);
a=beta(2:4);
A=[beta(8) beta(5)/2 beta(6)/2
beta(5)/2 beta(9) beta(7)/2
beta(6)/2 beta(7)/2 beta(10)];
```

The quadratic matric is:

```
disp(A)
```

```
    0.6089    0.3618    0.0305
    0.3618    0.9267    0.1337
    0.0305    0.1337    1.0213
```

Then is is possible to use the user-defined function *analyse_canonique()* provided in Moodle which computes the fix point and the eigen vectors $\vec{V}_i$ and eigen values λ_i :

```
[Xs,Ys,V,lamdda]=analyse_canonique(ao,a,A);
```

```
Xs = 3×1
   -0.9577
    0.3647
   -0.5120
Ys =
0.1602
V = 3×3
    0.8282    0.3394    0.4459
   -0.5553    0.3905    0.7342
    0.0751   -0.8558    0.5119
D = 3×3
    0.3690         0         0
         0    0.9481         0
         0         0    1.2397
```

The fix point X_s is

```
disp(array2table(Xs',"VariableNames",{'x_1','x_2','x_3'}))
```

x_1	x_2	x_3
-0.95766	0.36469	-0.51198

that we want to check to know if it is placed inside the domain.

The eigen vectors are:

```
disp(array2table(V,"VariableNames",{'V_1','V_2','V_3'}))
```

V_1	V_2	V_3
0.82822	0.33939	0.44594
-0.55535	0.39048	0.73424
0.075063	-0.85577	0.51189

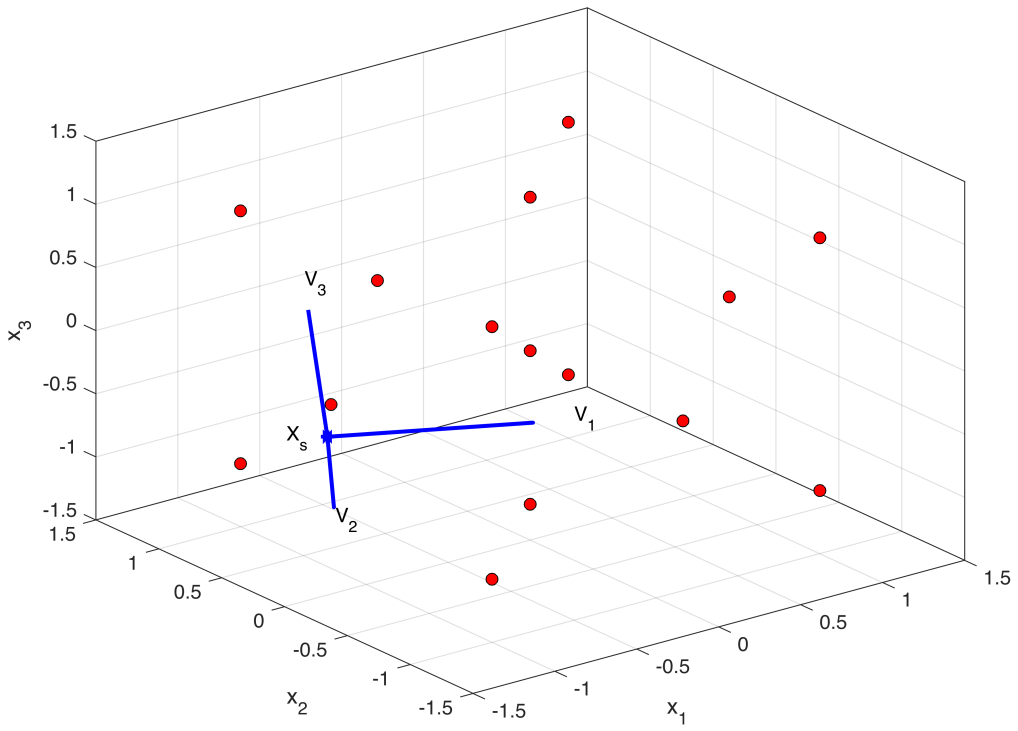
The eigen values are

```
disp(array2table(diag(lamdda)',"VariableNames",
{'lambda_1','lambda_2','lambda_3'}))
```

lambda_1	lambda_2	lambda_3
0.36904	0.94813	1.2397

Now it is possible to draw the position of the new reference frame (in red the experimental points are also indicated)

```
% Setup of the new referential frame
base=[Xs,Xs+V(:,1),Xs,Xs+V(:,2),Xs,Xs+V(:,3)];

figure
plot3(Xs(1),Xs(2),Xs(3),'*',base(1,:),base(2,:),base(3,:),'-',...
"LineWidth",2,"Color","blue")
hold on
plot3(E(:,1),E(:,2),E(:,3),'ok',...
"MarkerFaceColor","red")
hold off
box on
xlabel('x_1')
ylabel('x_2')
zlabel('x_3')
grid on

% coordinates of the labels of the eigen vectors
dl=1.2;
coord=[Xs-0.2*V(:,1),Xs+dl*V(:,1),Xs+dl*V(:,2),Xs+dl*V(:,3)]';
text(coord(:,1),coord(:,2),coord(:,3),["X_s","V_1","V_2", "V_3"])
```



Visualisation of the second degree fonction

Compute the value of the fonction in a parallelepiped around the origin

```
y_hat=beta'*f
```

```
y_hat =
```

$$\frac{342764994896069\,x_1^2}{562949953421312}+\frac{814722135220481\,x_1\,x_2}{1125899906842624}+\frac{275145962563595\,x_1\,x_3}{4503599627370496}+\frac{8408848579874081\,x_1}{9007199254740992}+\frac{208}{2}$$

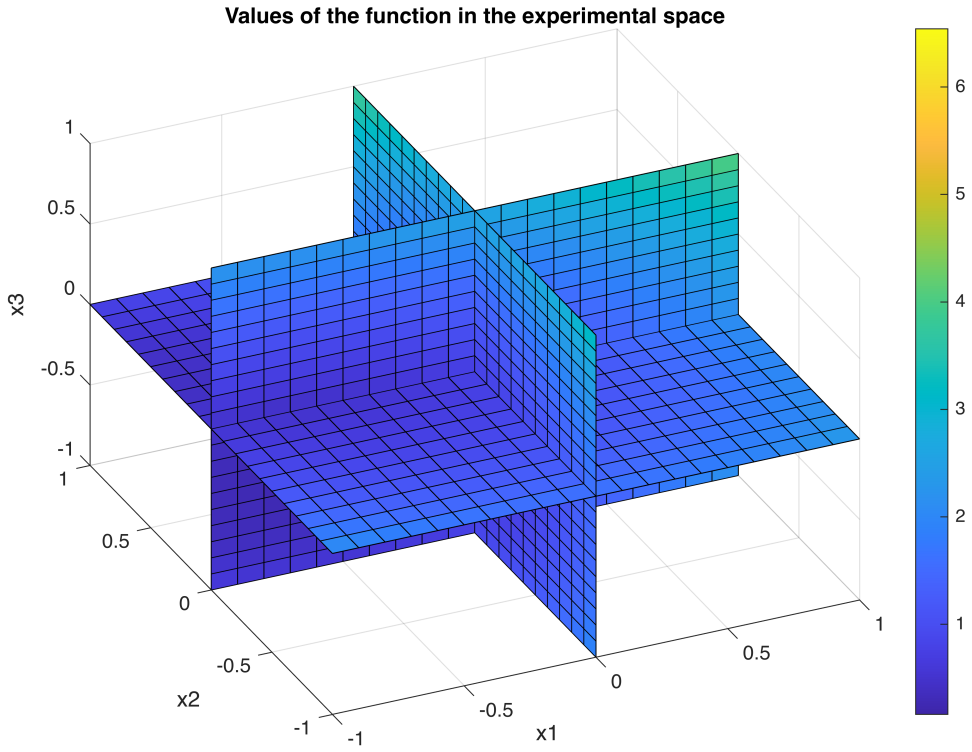
4) Computing values of the model on a grid

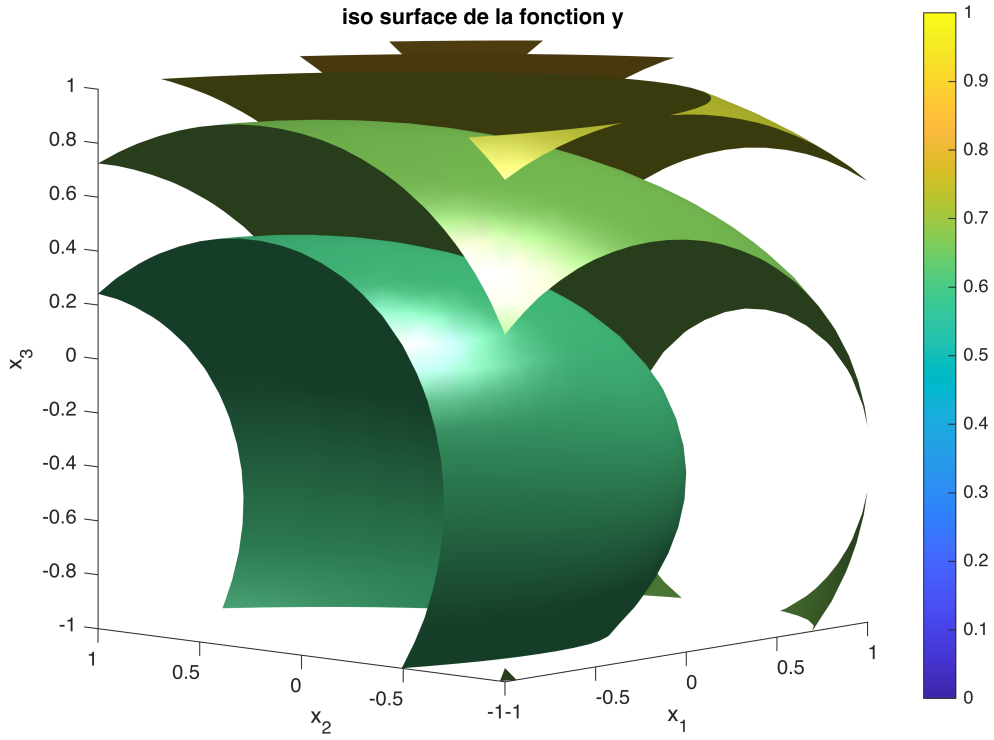
```
% number of points per dimension
N=21;

% domain
min_x=-1;
max_x=1;

% factor values
[X1,X2,X3]=meshgrid(linspace(min_x,max_x,N));
Xs=zeros(1,3);

% evaluation of the model
```

```
Y_hat=double(subs(y_hat,{x_1,x_2,x_3},{X1,X2,X3}));
```

5) Visualization

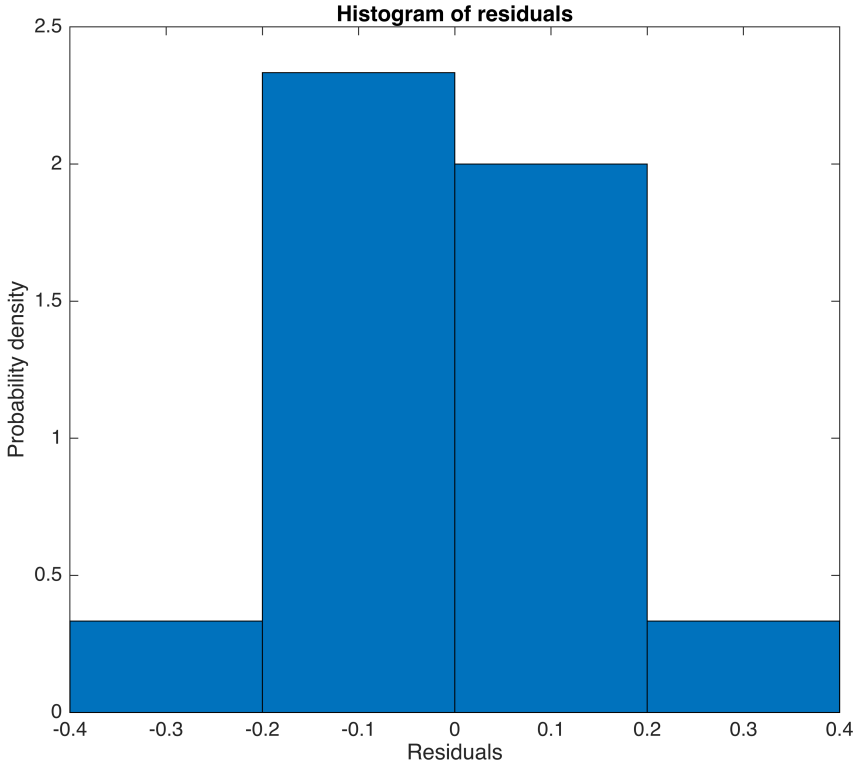
```
viz(X1,X2,X3,Y_hat,Xs)
```



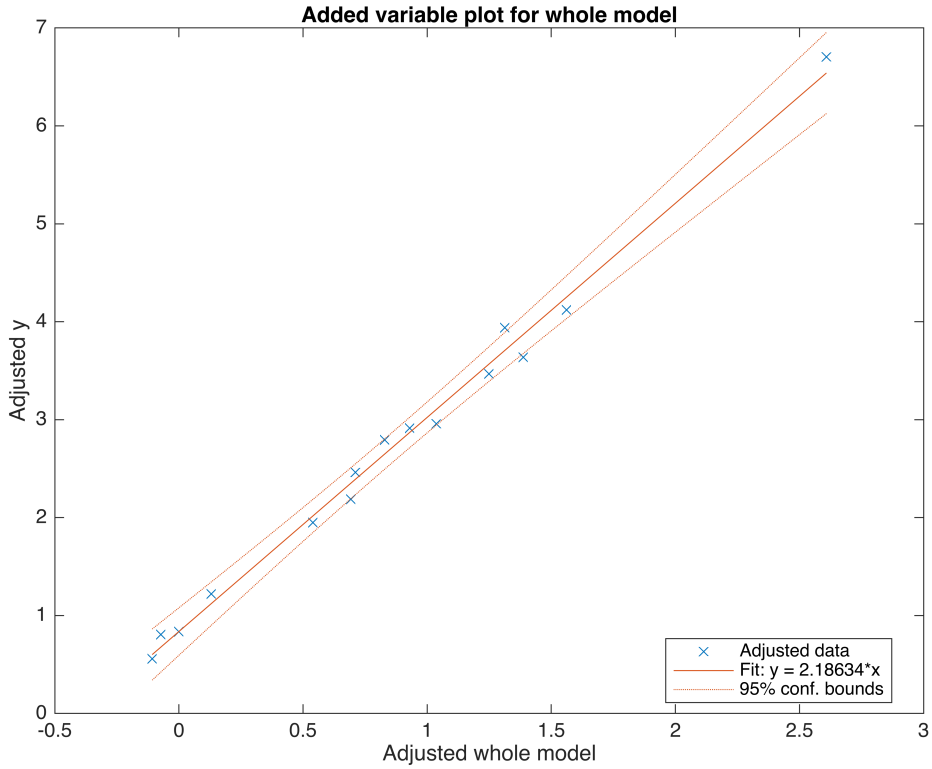


6) analyse the residuals

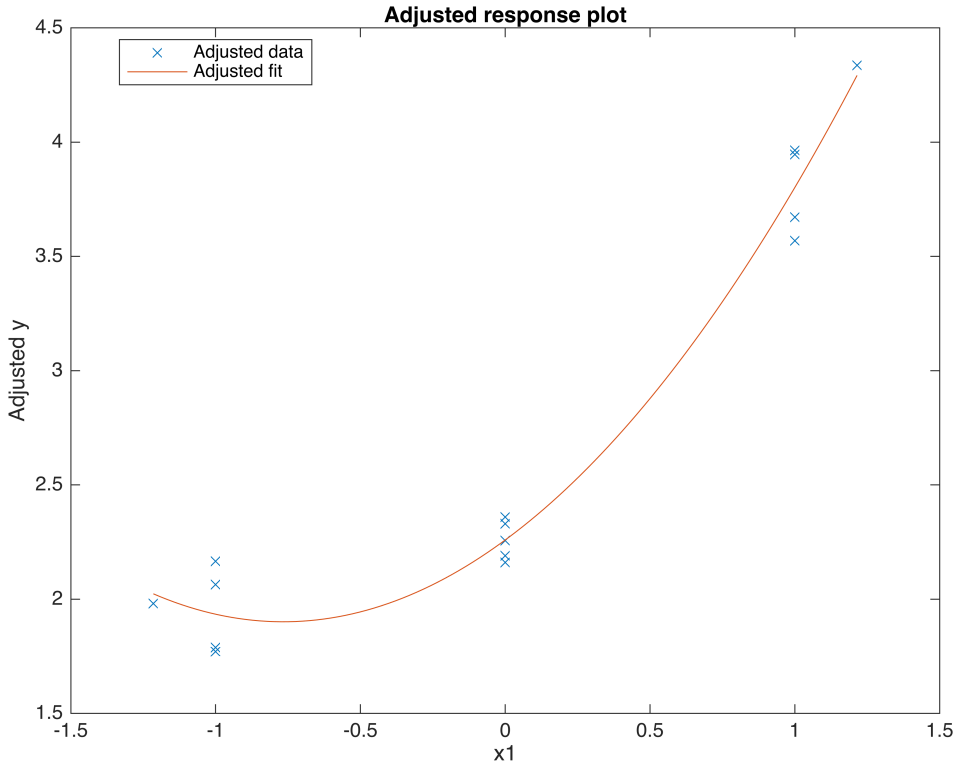
```
plotResiduals(mdl_1)
```



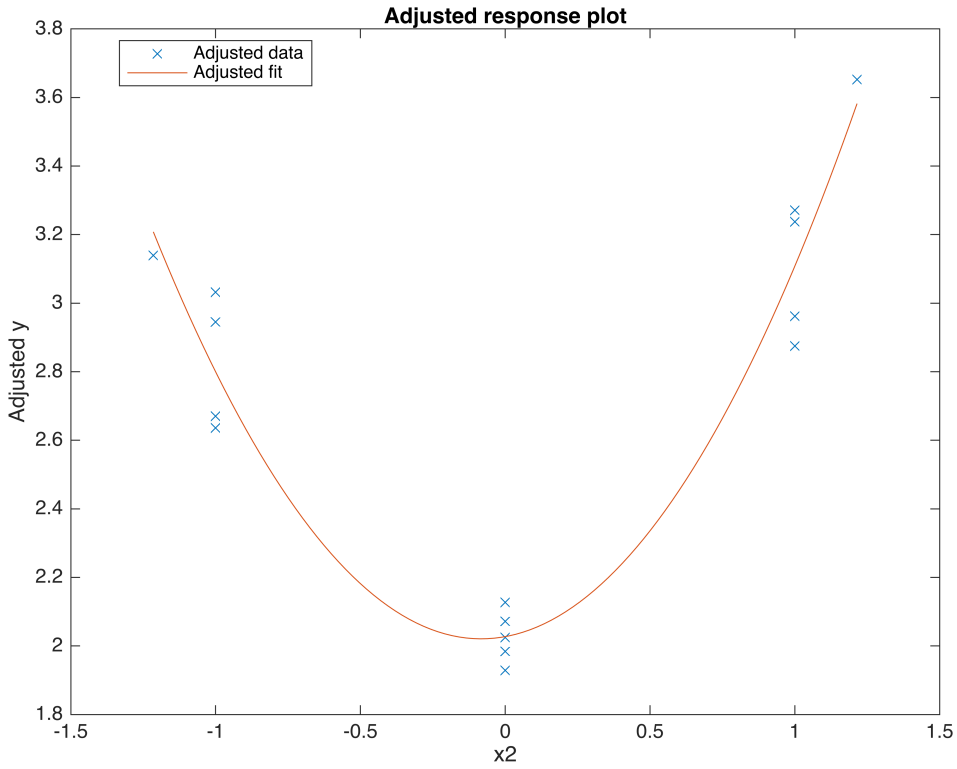

```
plotAdded(mdl_1)
```



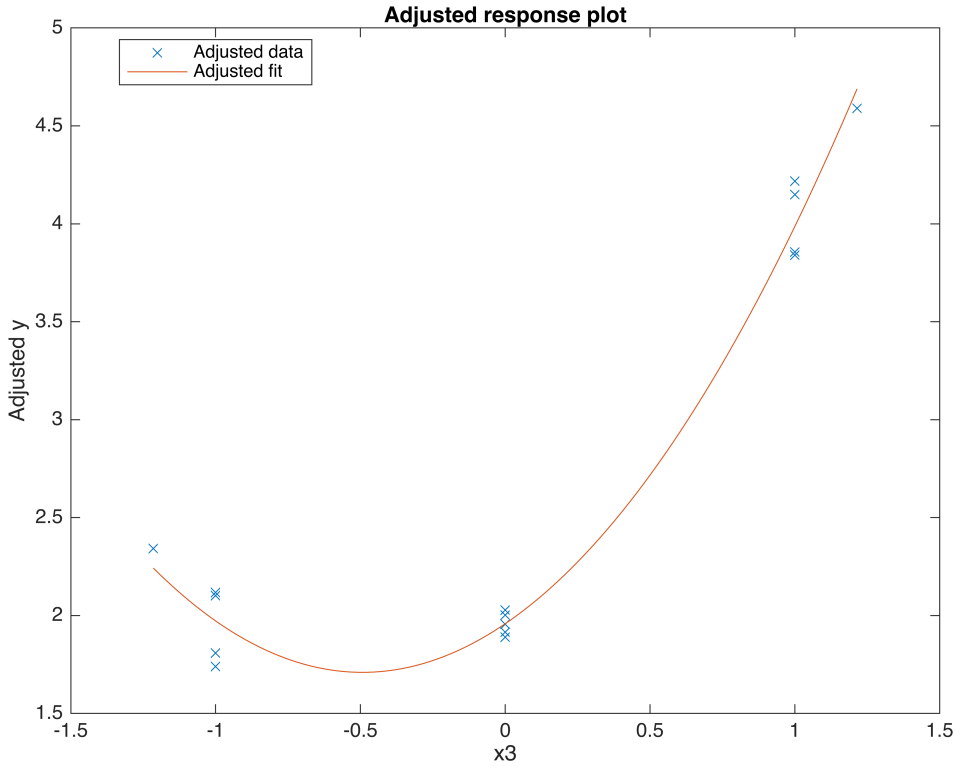
```
plotAdjustedResponse(mdl_1,"x1")
```




```
plotAdjustedResponse(mdl_1,"x2")
```




```
plotAdjustedResponse(mdl_1,"x3")
```



Strategy 2: Doehlert design

Let's see how the same situation results when using a Doehlert design

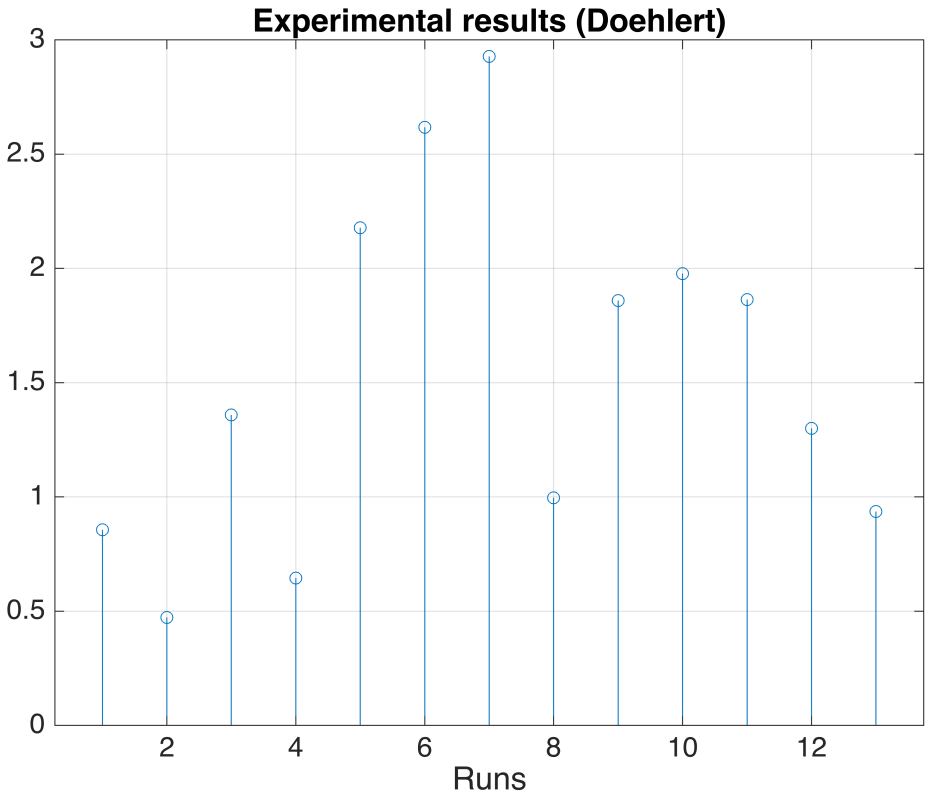
```
E_doehlert=doehlert(3);
disp(array2table(E_doehlert,"VariableNames",{'X_1','X_2','X_3'}))
```

X_1	X_2	X_3
0	0	0
-1	0	0
-0.5	-0.86603	0
-0.5	-0.28868	-0.8165
1	0	0
0.5	0.86603	0
0.5	0.28868	0.8165
-0.5	0.86603	0
-0.5	0.28868	0.8165
0	-0.57735	0.8165
0.5	-0.86603	0
0.5	-0.28868	-0.8165
0	0.57735	-0.8165

```
M_doehlert=x2fx(E_doehlert,"quadratic");
Nexp=size(M_doehlert,1); % number of experiments
Y_doehlert = M_doehlert*kappa'.*(ones(Nexp,1)+randn(Nexp,1)*sigma);
```

Graphical representation of the experimental results

```
figure
stem(Y_doehlert)
set(gca,'FontSize',14)
title('Experimental results (Doehlert)','FontSize',16)
xlabel('Runs','FontSize',16)
grid on
```




```
mdl_2=fitlm(E_doehlert,Y_doehlert,"quadratic")
```

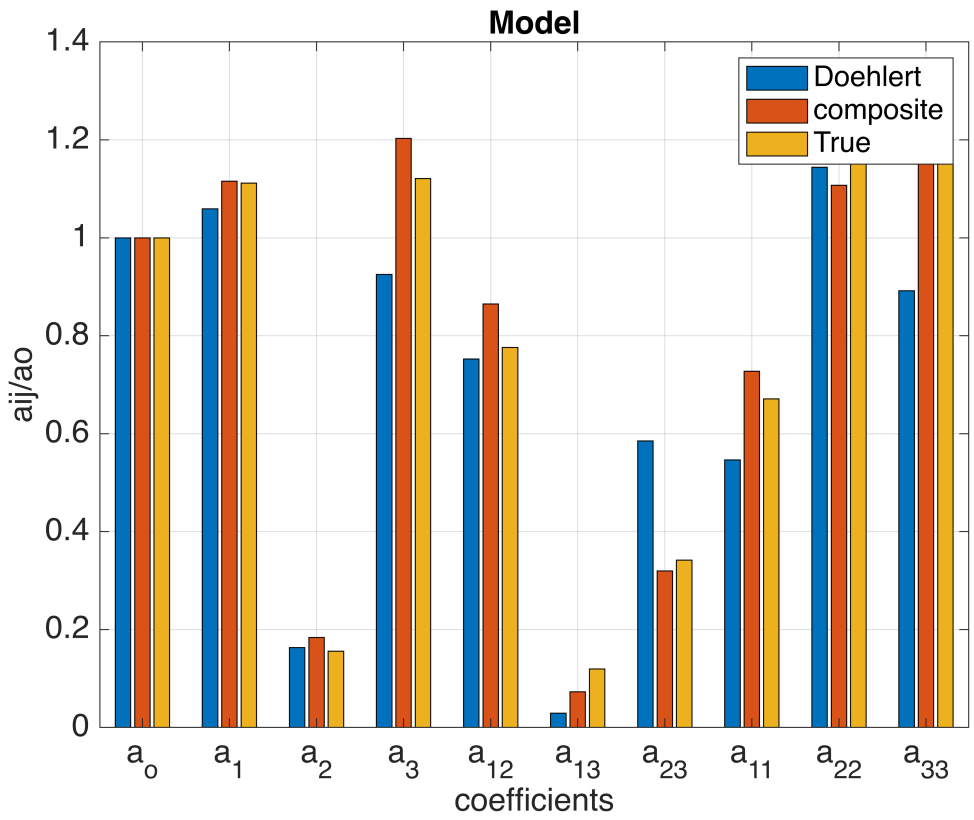
```
mdl_2 =
Linear regression model:
    y ~ 1 + x1*x2 + x1*x3 + x2*x3 + x1^2 + x2^2 + x3^2

Estimated Coefficients:
                   Estimate       SE        tStat       pValue
                   ________    ________    _______    __________

    (Intercept)     0.85687     0.11653     7.3533     0.0051979
    x1               0.9075    0.058264     15.576    0.00057507
    x2              0.13963    0.058264     2.3965      0.096176
    x3              0.79267    0.058264     13.605    0.00085904
    x1:x2           0.64483     0.13456     4.7923      0.017283
    x1:x3          0.025054     0.15044    0.16654       0.87833
    x2:x3           0.50162     0.15044     3.3344      0.044572
    x1^2            0.46839     0.14272     3.2819      0.046359
    x2^2            0.98053     0.14272     6.8704     0.0063148
    x3^2             0.7644      0.1387     5.5113      0.011762


Number of observations: 13, Error degrees of freedom: 3
Root Mean Squared Error: 0.117
R-squared: 0.994,  Adjusted R-Squared: 0.977
F-statistic vs. constant model: 57.4, p-value = 0.00337
```

```
anova(mdl_2,"summary",2)
```

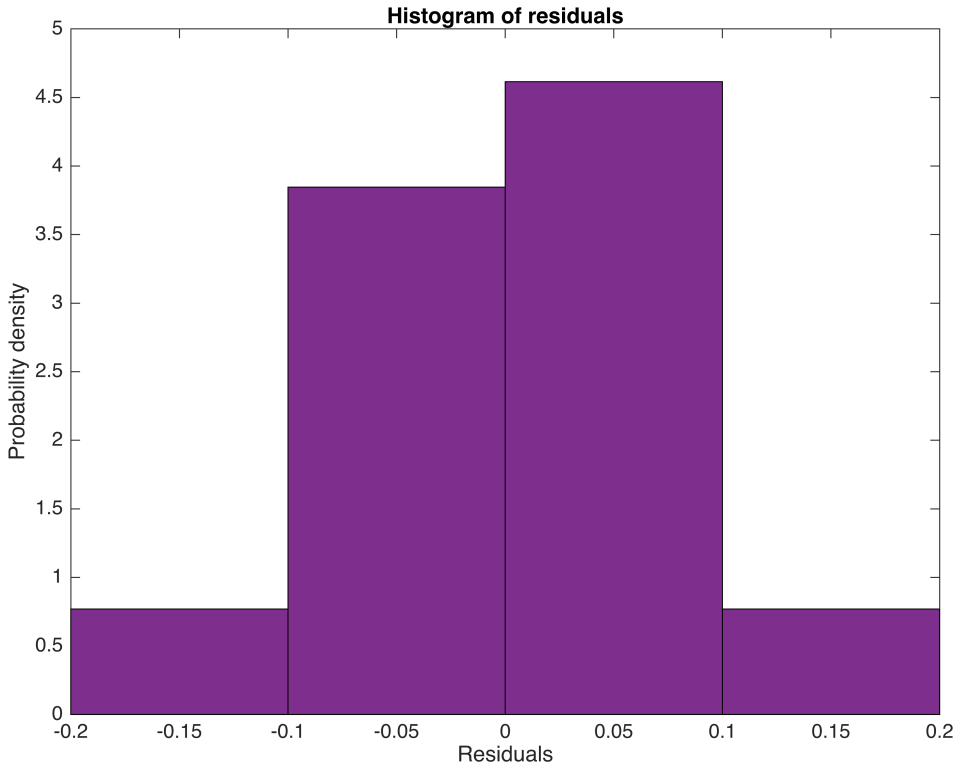
```
ans = 5×5 table
```

	SumSq	DF	MeanSq	F	pValue
1 Total	7.0506	12	0.5876	NaN	NaN
2 Model	7.0099	9	0.7789	57.3594	0.0034
3 . Linear	5.8856	3	1.9619	144.4782	9.6550e-04
4 . Nonlinear	1.1243	6	0.1874	13.7999	0.0274
5 Residual	0.0407	3	0.0136	NaN	NaN

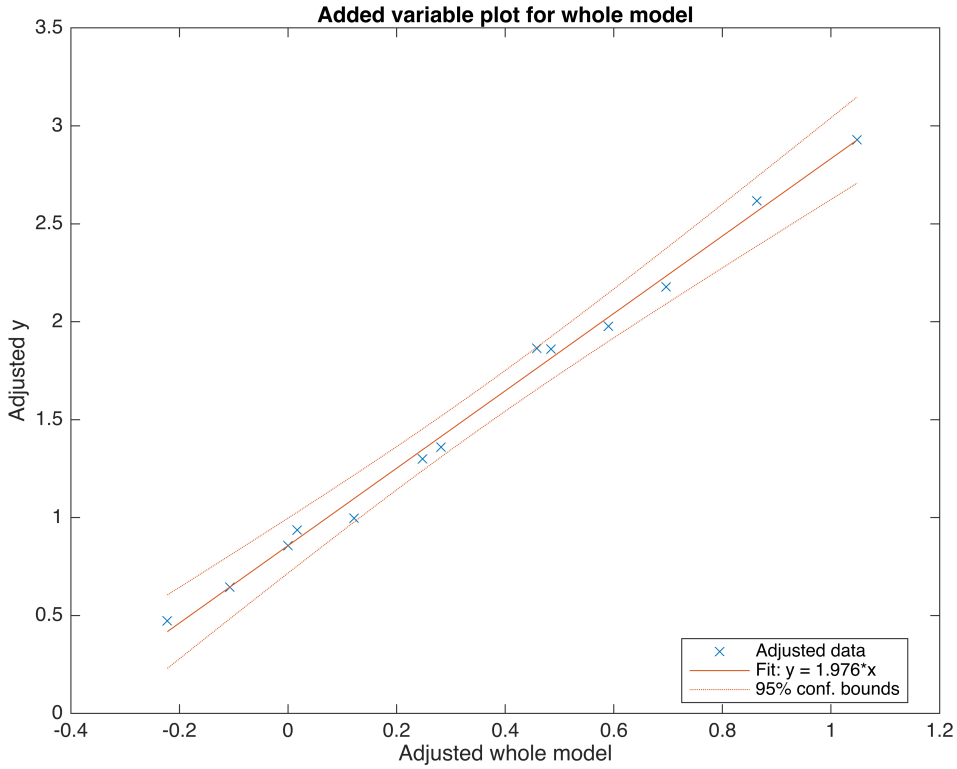
```
figure
gamma=mdl_2.Coefficients.Estimate;
bar([gamma/gamma(1), beta/beta(1), kappa'/kappa(1)])
set(gca,'FontSize',16,...
    'XTickLabel',{'a_o' 'a_1' 'a_2' 'a_3' ...
    'a_{12}' 'a_{13}' 'a_{23}' 'a_{11}' 'a_{22}' 'a_{33}'})
title('Model','FontSize',16)
xlabel('coefficients','FontSize',16)
ylabel('aij/ao','FontSize',16)
legend("Doehlert","composite", "True")
grid on
```



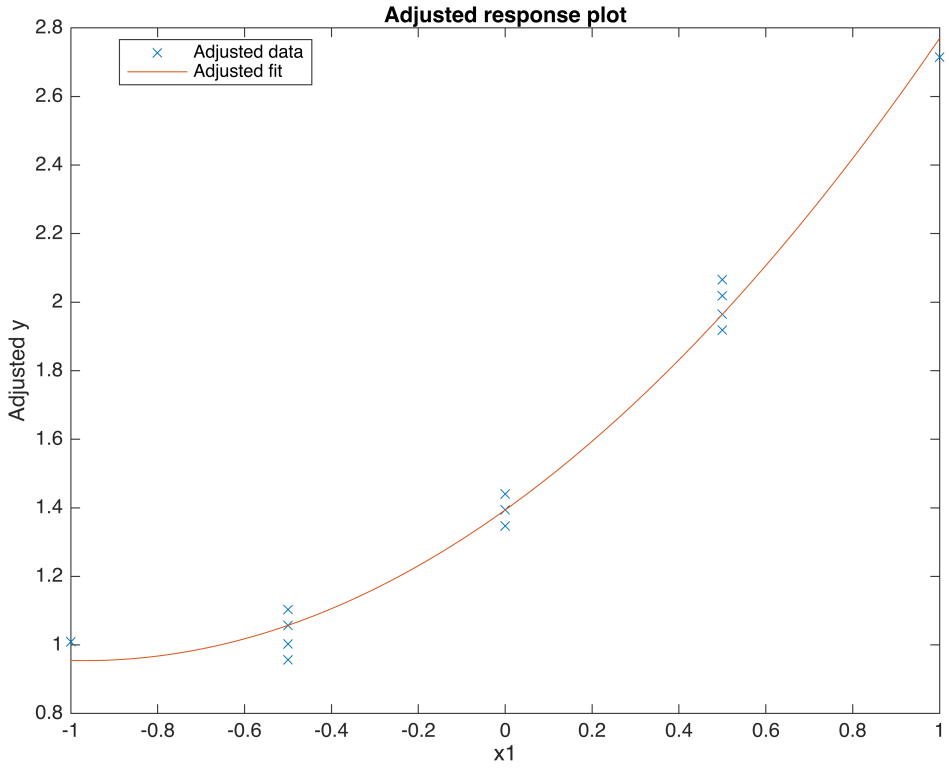

```
plotResiduals(mdl_2)
```



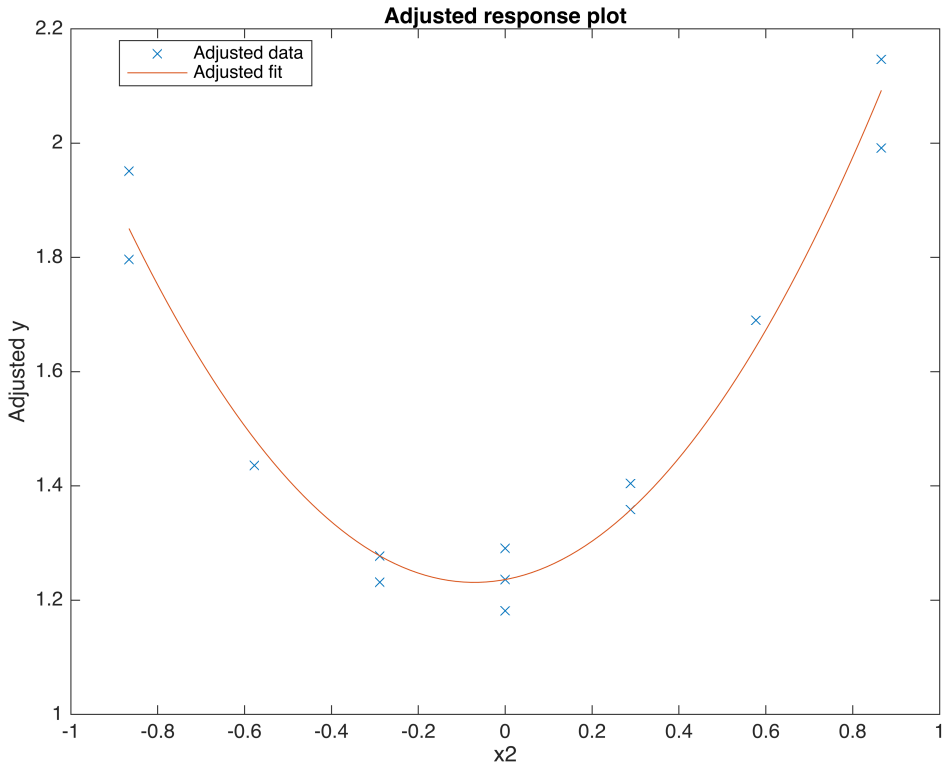

```
plotAdded(mdl_2)
```



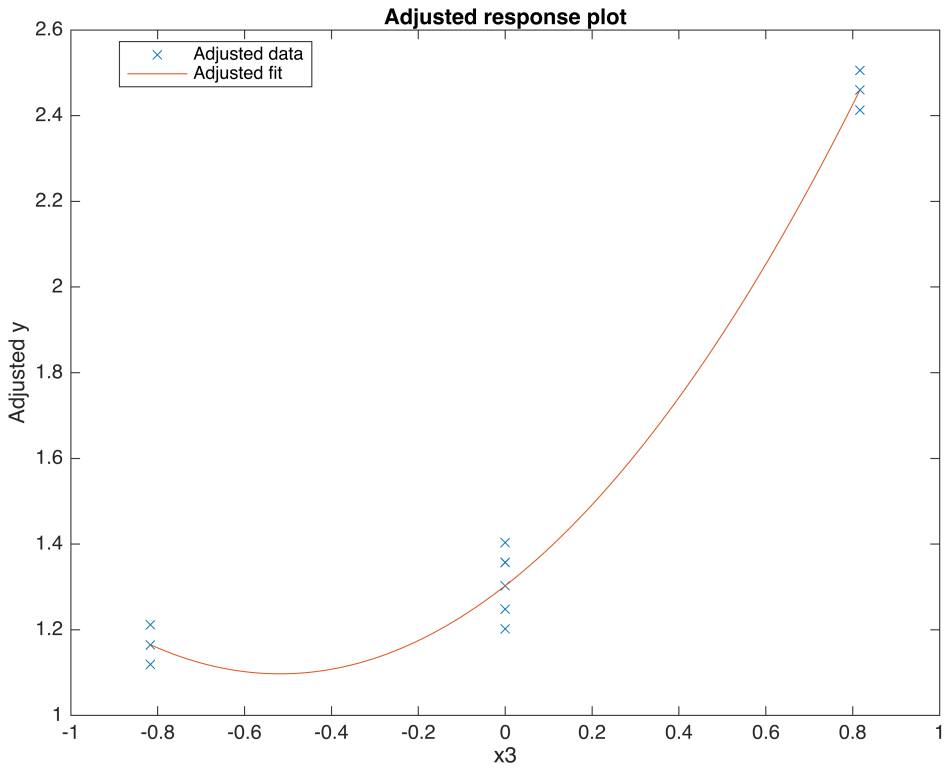
```
plotAdjustedResponse(mdl_2,"x1")
```




```
plotAdjustedResponse(mdl_2,"x2")
```




```
plotAdjustedResponse(mdl_2,"x3")
```



Strategy 3 : Box-Behnken design

Let's see how the same situation results when using a Doehlert design

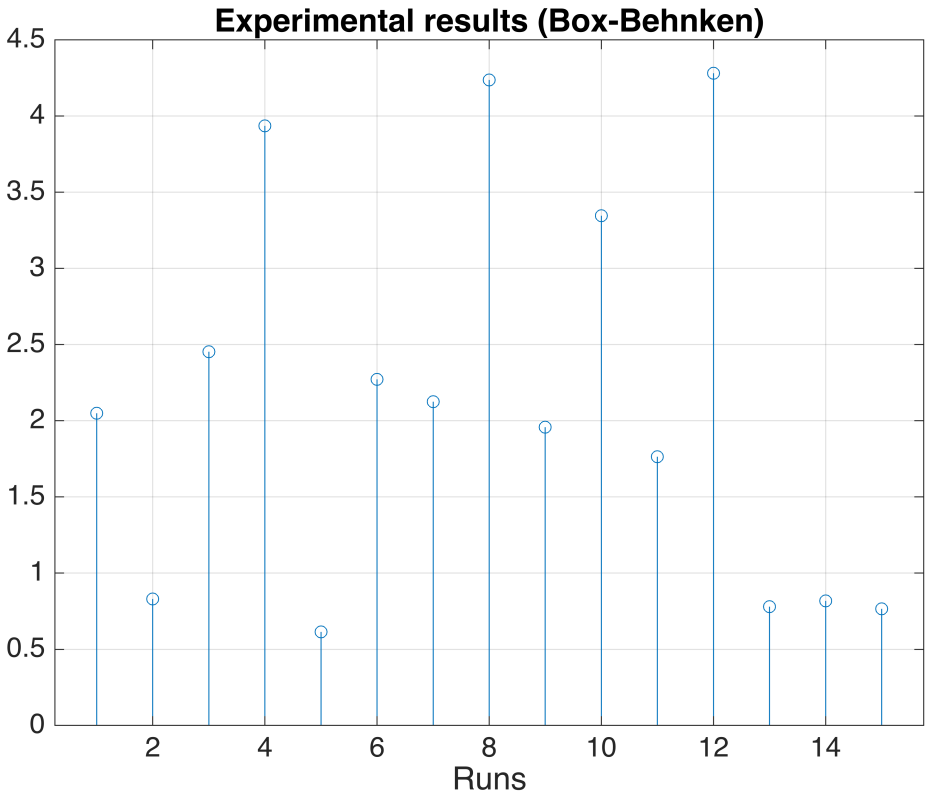
```
E_BB=bbdesign(3,"center",3);
disp(array2table(E_BB,"VariableNames",{'X_1','X_2','X_3'}))
```

X_1	X_2	X_3
-1	-1	0
-1	1	0
1	-1	0
1	1	0
-1	0	-1
-1	0	1
1	0	-1
1	0	1
0	-1	-1
0	-1	1
0	1	-1
0	1	1
0	0	0
0	0	0
0	0	0

```
M_BB=x2fx(E_BB,"quadratic");
Nexp=size(M_BB,1); % number of experiments
Y_BB = M_BB*kappa'.*(ones(Nexp,1)+randn(Nexp,1)*sigma);
```

Graphical representation of the experimental results

```
figure
stem(Y_BB)
set(gca,'FontSize',14)
title('Experimental results (Box-Behnken)','FontSize',16)
xlabel('Runs','FontSize',16)
grid on
```




```
mdl_3=fitlm(E_BB,Y_BB,"quadratic")
```

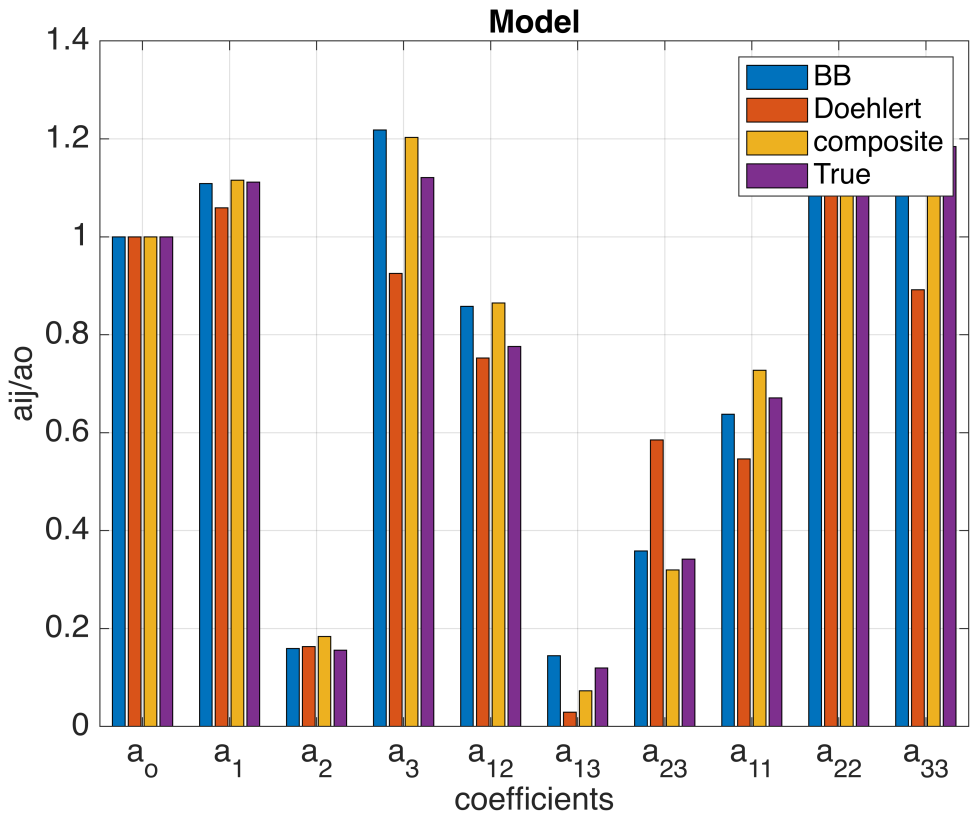
```
mdl_3 =
Linear regression model:
    y ~ 1 + x1*x2 + x1*x3 + x2*x3 + x1^2 + x2^2 + x3^2

Estimated Coefficients:
                   Estimate       SE       tStat       pValue
                   ________    ________    ______    __________

    (Intercept)    0.78756     0.046336    16.997    1.2898e-05
    x1              0.8731     0.028375     30.77     6.804e-07
    x2             0.12526     0.028375    4.4143     0.0069286
    x3             0.95921     0.028375    33.805    4.2594e-07
    x1:x2          0.67577     0.040128     16.84    1.3498e-05
    x1:x3          0.11363     0.040128    2.8316      0.036604
    x2:x3          0.28235     0.040128    7.0363     0.0008952
    x1^2           0.50213     0.041767    12.022    7.0262e-05
    x2^2            1.0277     0.041767    24.605    2.0678e-06
    x3^2            1.0224     0.041767    24.478    2.1219e-06


Number of observations: 15, Error degrees of freedom: 5
Root Mean Squared Error: 0.0803
R-squared: 0.999,  Adjusted R-Squared: 0.996
F-statistic vs. constant model: 405, p-value = 1.29e-06
```

```
anova(mdl_3,"summary",2)
```

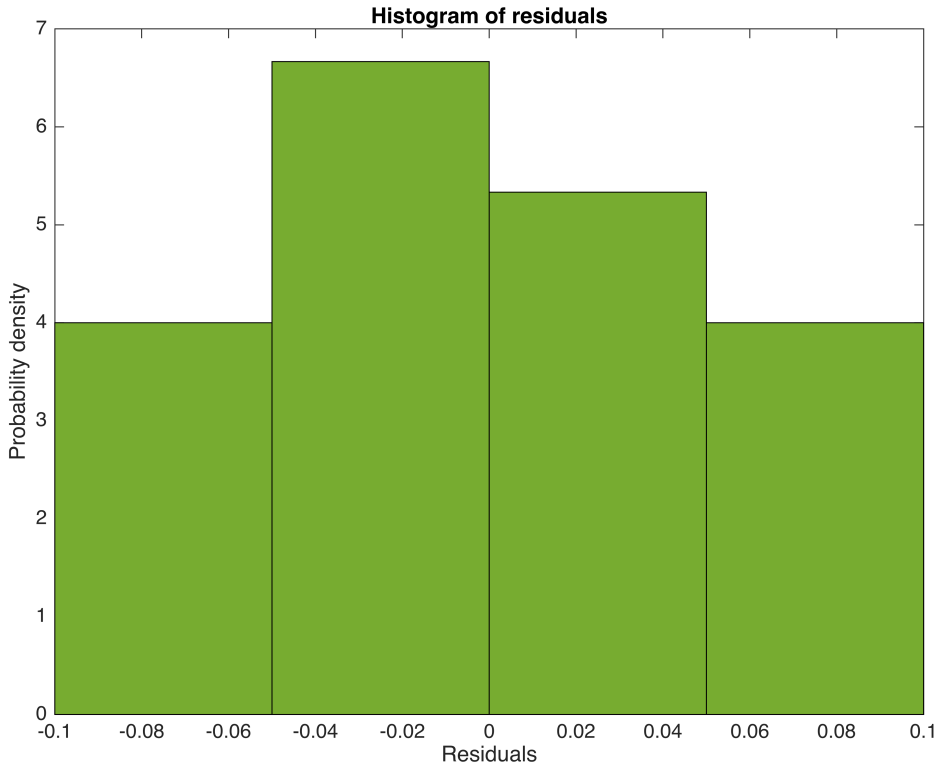
```
ans = 7×5 table
```

	SumSq	DF	MeanSq	F	pValue
1 Total	23.4911	14	1.6779	NaN	NaN
2 Model	23.4589	9	2.6065	404.6715	1.2918e-06
3 . Linear	13.5846	3	4.5282	703.0166	5.5373e-07
4 . Nonlinear	9.8742	6	1.6457	255.4990	4.7236e-06
5 Residual	0.0322	5	0.0064	NaN	NaN
6 . Lack of fit	0.0307	3	0.0102	13.8369	0.0681
7 . Pure error	0.0015	2	7.4017e-04	NaN	NaN

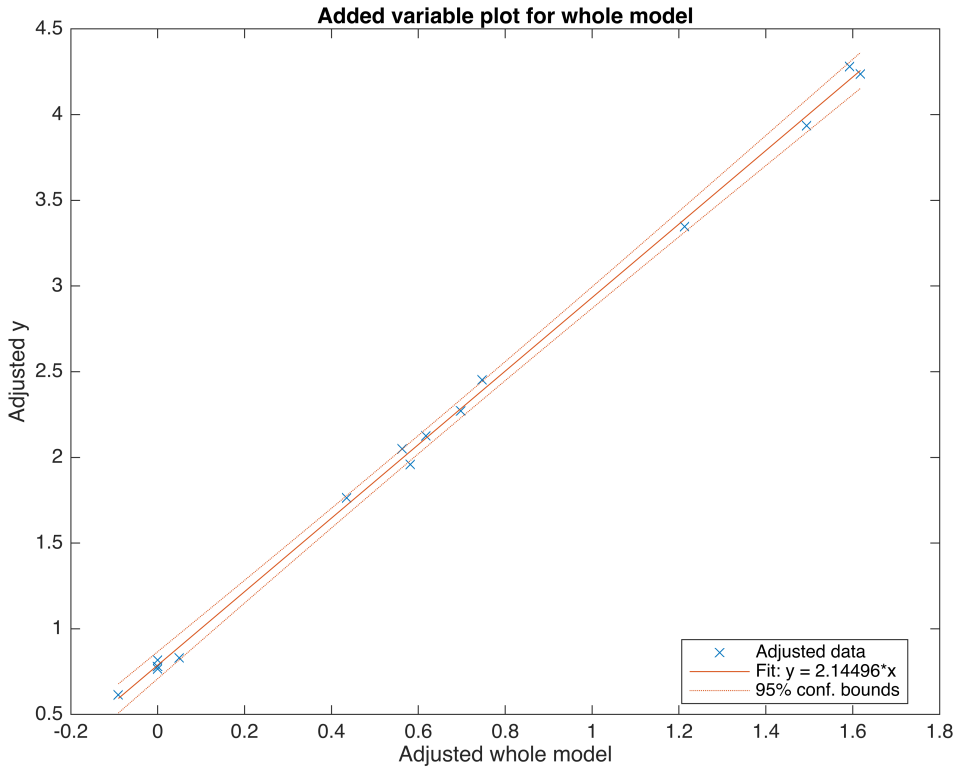
```
figure
iota=mdl_3.Coefficients.Estimate;
bar([iota/iota(1),gamma/gamma(1), beta/beta(1), kappa'/kappa(1)])
set(gca,'FontSize',16,...
    'XTickLabel',{'a_o' 'a_1' 'a_2' 'a_3' ...
    'a_{12}' 'a_{13}' 'a_{23}' 'a_{11}' 'a_{22}' 'a_{33}'})
title('Model','FontSize',16)
xlabel('coefficients','FontSize',16)
ylabel('aij/ao','FontSize',16)
legend("BB","Doehlert","composite", "True")
grid on
```



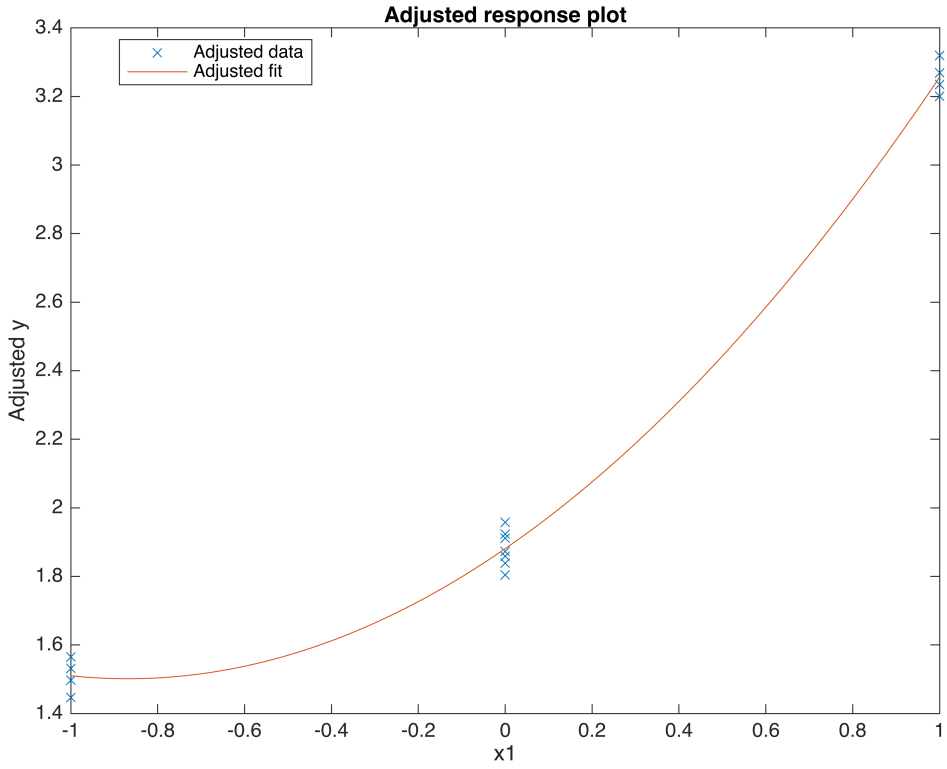
```
plotResiduals(mdl_3)
```



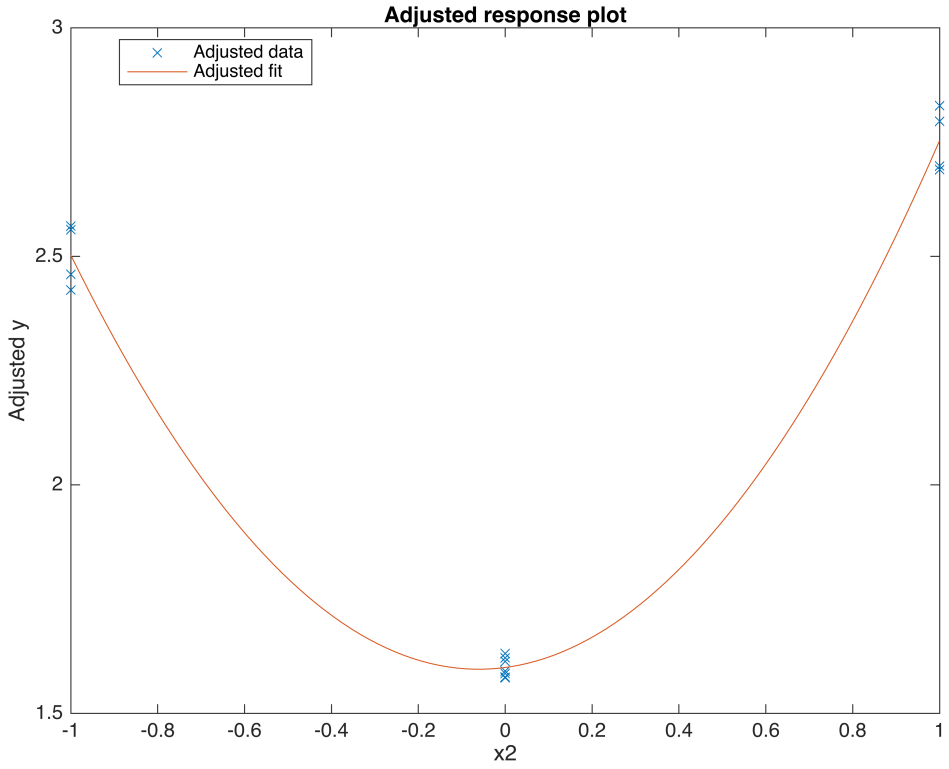

```
plotAdded(mdl_3)
```



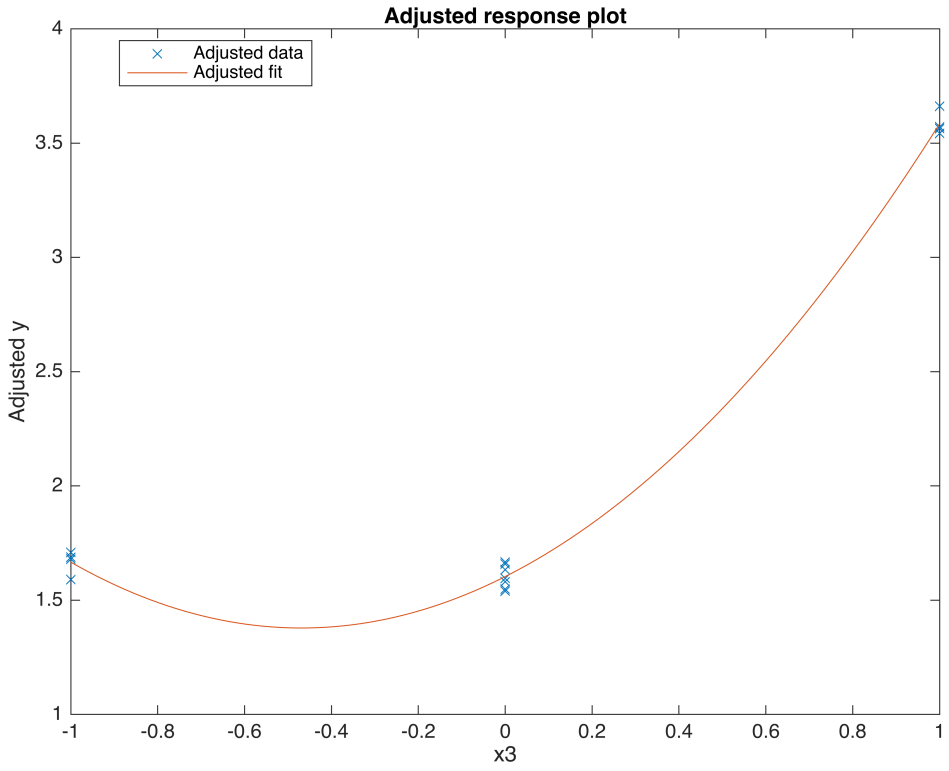

```
plotAdjustedResponse(mdl_3,"x1")
```



```
plotAdjustedResponse(mdl_3,"x2")
```




```
plotAdjustedResponse(mdl_3,"x3")
```




```
function viz(X1,X2,X3,Y,Xs)
% viz(ValuesMatrix,X_s)
% Y: matrix with the values of the function
% Xi: factors values
```

The 3 planes go by x_s.

```
slice(X1,X2,X3,Y,Xs(1),Xs(2),Xs(3))
title('Values of the function in the experimental space')
xlabel('x1')
ylabel('x2')
zlabel('x3')
colorbar
view([-24.70 40.40])
```

Isosurface drawing

```
% save the colormap for defining automatiquely the color of the surface
map=colormap;
```

```
% Compute the limit of Y
Ymin=min(min(min(Y)));
Ymax=max(max(max(Y)));

% number of surfaces
Ns=7;

% define the levels of the isosurfaces
Levels= linspace(Ymin,Ymax,Ns);

figure


for k=2:6
    % compute the iso surface and built the patches
    p=patch(isosurface(X1,X2,X3,Y,Levels(k)));
    isonormals(X1,X2,X3,Y,p)

    % define the color of the surface from the actual color map
    p.FaceColor=map(round(120+120/7*k),1:3);
    p.EdgeColor='none';
    hold on
end

colorbar

% close the drawing loop
hold off

% define a cubique zone for the 3D la representation
daspect([1 1 1])
%view(3);
view([-48 37])
axis tight
camlight
lighting gouraud % lighting adéquat pour surface courbée
view([-48.30 8.40])
title('iso surface de la fonction y')
xlabel('x_1')
ylabel('x_2')
zlabel('x_3')


end
```