```
*************************************************************************;
*** This Program finds optimal survey designs in the study of       **;
*** of the level effect, using a simple exchange algorithm          **;
*************************************************************************;

options  ls=max ps=max nocenter notes;
libname mylib 'U:\data';
proc iml;

%** Input the number of levels for each factor. For example, if there **;
%** are 5 factors with (2,3,4,2,3) levels, define factors={2,3,4,2,3} **;
%** The # of levels can only be 2, 3, or 4.                           **;
factors={4,2,3};

%** Number of trials                                                  **;
n_trials=12;


%** Specify d/a ratio in Lambda                              **;
aa=1;
*bb=0.1;*bb=0.2;*bb=0.25;
***************;
*bb=0.3;*bb=0.4;*bb=0.5;*bb=1;*bb=1.25;
***************;
*bb=1.5; *bb=4;
bb=2;
***************;
*bb=4.5;*bb=5;*bb=10;*bb=15;
***************;
*bb=50;
***************;
*bb=-0.1;
***************;
*bb=-0.2;*bb=-0.25;*bb=-0.3;
***************;
*bb=-0.4;*bb=-0.455;
***************;
*bb=-0.495;

%** Specify max number of iterations                                 **;
max_iteration = 1000;


%** Are there initial designs (1=yes 0=no)?                          **;
has_init=0;
```

```
%** Initial design (needed if has_init=1)                              **;
**init=;
%** Specify random seed to generate random initial designs            **;
*seed=1025;
*seed=0601200509;
seed=614292;
*seed=579;
%** Specify number of initial trials to be generated randomly         **;
n_init = 3;


%** number of parameters (not including the intercept)     **;
p=sum(factors) - nrow(factors) + 1;
JM=j(p,p,1);

JM1=j(p-2,p-2,1);
a1=aa;b1=bb;

%*********************************************************;
%** input matrix RR1                                     **;
%** from the first study, fixed X1,Z1                    **;
%** Use standardized orthogonal coding                   **;
%*********************************************************;

%** Transformation Matrix from the standardized orthogonal **;
%** effects coding to dummy coding of X1                    **;

TT1={   1   0.5    0.5    0.33    0.33,
        0   0.5    0.0    0.00    0.00,
        0   0.0   -0.5    0.00    0.00,
        0   0.0    0.0    0.00   -0.41,
        0   0.0    0.0    0.47   -0.24};

%** X1 matrix under the dummy coding scheme               **;

X1={1   0 1 0 1,
    1  0 1 1 0,
    1 0 1 0 0,
    1 0 0 0 1,
    1 0 0 1 0,
    1 0 0 0 0,
    1 1 1 0 1,
    1 1 1 1 0,
    1 1 1 0 0,
    1 1 0 0 1,
    1 1 0 1 0,
```

```
    1 1 0 0 0};

%** X1 matrix under the standardized orthogonal effects coding    **;

X1=X1*inv(TT1);

%** Z1 matrix under the dummy coding of X1                 **;

Z1={1 0 0 0 0 0 0 0,
    0 1 2 0 0 0 0 0,
    0 0 0 0 0 1 0 0,
    0 0 0 0 0 0 1 0,
    0 0 0 0 0 0 0 1};

%** Z1 matrix under the standardized orthogonal effects coding    **;

Z1=TT1*Z1;


RR1=Z1‘*X1‘*X1*inv(i(p-2)+a1*X1‘*X1+b1*JM1*X1‘*X1)*Z1;

%*********************************************************;
%** Z2 intrix from the second study                      **;
%*********************************************************;

%** Transformation Matrix from the standardized orthogonal **;
%** effects coding to dummy coding of X2                  **;

TT2={   1   0.25   0.25   0.25    0.5    0.33    0.33,
        0   0.00   0.00   0.35    0.0    0.00    0.00,
        0   0.00   0.41  -0.20    0.0    0.00    0.00,
        0   0.43  -0.14  -0.14    0.0    0.00    0.00,
        0   0.00   0.00   0.00   -0.5    0.00    0.00,
        0   0.00   0.00   0.00    0.0    0.00   -0.41,
        0   0.00   0.00   0.00    0.0    0.47   -0.24};

%** Z2 matrix under the dummy coding scheme of X2      **;


Z2={  1   0     0      0      0     0   0   0,
      0   0.33  1.33  -0.11  -0.44  0   0   0,
      0   0.67  2.67   0.22   0.88  0   0   0,
      0   1     4      0      0     0   0   0,
      0   0     0      0      0     1   0   0,
      0   0     0      0      0     0   1   0,
      0   0     0      0      0     0   0   1};
```

```
%** Z2 matrix under the standardized orthogonal coding **;
ZM=TT2*Z2;



%** DT the dg(theta)/d(theta) derivative               **;
%** Assuming k=3,v=0.2,w=0.5                          **;

DT={    1    0    0    0       0    0    0,
        0    1    0    0       0    0    0,
        0    0    1    0       0    0    0,
        0    2    0    0.33    0    0    0,
        0    0    2    5.0     0    0    0,
        0    0    0    0       1    0    0,
        0    0    0    0       0    1    0,
        0    0    0    0       0    0    1};


%************************************************************;
%* coefficients of standardized orthogonal contrasts         **;
%* for factors with 2,3 and 4 levels. This program is        **;
%* limited to factors that contain less than 4 levels        **;
%* it can be modified to accommodate factors with more levels**;
%************************************************************;

d2={1,
   -1};

d3={1.22  -0.71,
    0      1.41,
   -1.22  -0.71};

d4={1.41  -0.82  -0.58,
    0      1.63  -0.58,
    0      0      1.73,
   -1.41  -0.82  -0.58};

%************************************************************;
%* Create all candidate design points x                     **;
%************************************************************;

num_factors = nrow(factors);

if num_factors > 10 then do;
```

```
  print "ERROR: NUMBER OF FACTORS EXCEEDS 10 !!!";
  stop;
end;

factors_exp = factors;
do i=1 to 10-num_factors;
  factors_exp = factors_exp // {1};
end;

levels=j(10,1,0);

do i=1 to 10;
  levels[i] = factors_exp[i,1];
end;

if levels[1]=2 then m1=d2;
  else if levels[1]=3 then m1=d3;
  else if levels[1]=4 then m1=d4;
if levels[2]=2 then m2=d2;
  else if levels[2]=3 then m2=d3;
  else if levels[2]=4 then m2=d4;
if levels[3]=2 then m3=d2;
  else if levels[3]=3 then m3=d3;
  else if levels[3]=4 then m3=d4;
if levels[4]=2 then m4=d2;
  else if levels[4]=3 then m4=d3;
  else if levels[4]=4 then m4=d4;
if levels[5]=2 then m5=d2;
  else if levels[5]=3 then m5=d3;
  else if levels[5]=4 then m5=d4;
if levels[6]=2 then m6=d2;
  else if levels[6]=3 then m6=d3;
  else if levels[6]=4 then m6=d4;
if levels[7]=2 then m7=d2;
  else if levels[7]=3 then m7=d3;
  else if levels[7]=4 then m7=d4;
if levels[8]=2 then m8=d2;
  else if levels[8]=3 then m8=d3;
  else if levels[8]=4 then m8=d4;
if levels[9]=2 then m9=d2;
  else if levels[9]=3 then m9=d3;
  else if levels[9]=4 then m9=d4;
if levels[10]=2 then m10=d2;
  else if levels[10]=3 then m10=d3;
  else if levels[10]=4 then m10=d4;
```

```
%** number of candidate designs                          **;
cand_n = 1;
do i=1 to nrow(factors);
  cand_n=cand_n*factors[i];
end;

do i1 = 1 to levels[1];
do i2 = 1 to levels[2];
do i3 = 1 to levels[3];
do i4 = 1 to levels[4];
do i5 = 1 to levels[5];
do i6 = 1 to levels[6];
do i7 = 1 to levels[7];
do i8 = 1 to levels[8];
do i9 = 1 to levels[9];
do i10 = 1 to levels[10];
  x=m1[i1,];  d=i1;
  if levels[2]>1 then do;  x=x||m2[i2,]; d=d||i2;  end;
  if levels[3]>1 then do;  x=x||m3[i3,]; d=d||i3;  end;
  if levels[4]>1 then do;  x=x||m4[i4,]; d=d||i4;  end;
  if levels[5]>1 then do;  x=x||m5[i5,]; d=d||i5;  end;
  if levels[6]>1 then do;  x=x||m6[i6,]; d=d||i6;  end;
  if levels[7]>1 then do;  d=x||m7[i7,]; d=d||i7;  end;
  if levels[8]>1 then do;  d=x||m8[i8,]; d=d||i8;  end;
  if levels[9]>1 then do;  d=x||m9[i9,]; d=d||i9;  end;
  if levels[10]>1 then do; d=x||m10[i10,]; d=d||i10; end;

  if i1=1 & i2=1 & i3=1 & i4=1 & i5=1 &
     i6=1 & i7=1 & i8=1 & i9=1 & i10=1 then do;
       cand_x=x;
   cand_d=d;
  end;
  else do;
       cand_x=cand_x//x;
       cand_d=cand_d//d;
  end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
```

```
end;

cand_x= j(nrow(cand_x),1,1) || cand_x;

%** print cand_x, cand_d;

if nrow(cand_x) ^= cand_n then do;
  print "Error: number of candidate designs is incorrect!!";
  stop;
end;
if ncol(cand_x) ^= p then do;
  print "Error: number of parameters is incorrect!!";
  stop;
end;

%****************************************************************;
%* create initial design matrix                                 **;
%****************************************************************;

if has_init=1 then do;
  init_d=init;
  init_n=nrow(init);

  do i=1 to nrow(init_d);
    x={1};
do j=1 to ncol(init_d);
  k=init_d[i,j];
      if factors[j]=2 then x=x||d2[k,];
  if factors[j]=3 then x=x||d3[k,];
  if factors[j]=4 then x=x||d4[k,];
end;
if i=1 then init_x=x;
else init_x=init_x//x;
  end;
end;

%** print init_x init_d;

if has_init=0 then do;
  init_n = n_init;

  do i=1 to init_n;
    ran=ranuni(seed);
    index=ceil(ran*cand_n);

if i=1 then do;
```

```
      init_x=cand_x[index,];
  init_d=cand_d[index,];
end;
else do;
  init_x=init_x//cand_x[index,];
  init_d=init_d//cand_d[index,];
end;
  end;
end;

print "initial designs";
if has_init=1 then print "(input by user)";
if has_init=0 then print "(randomly generated)";
print init_x, init_d;

if init_n > n_trials then do;
  print "Error: number of initial designs exceeds the number of trials!!!";
  stop;
end;


%*************************************************************************;
%* sequentially add to n_trials                                          **;
%*************************************************************************;

design_x = init_x;
design_d = init_d;

do i=1 to n_trials - init_n ;
  do j=1 to cand_n;
    x=cand_x[j,];

        den_temp1 = design_x‘*design_x + x‘*x;
    den_temp2 = i(p)+ aa*design_x‘*design_x
                    + bb*JM*design_x‘*design_x + aa*x‘*x +bb*JM*x‘*x;
temp=DT‘*ginv(ZM‘*den_temp1*inv(den_temp2)*ZM + RR1)*DT;
    det_add = 1/(det(DT‘*ginv(ZM‘*den_temp1*inv(den_temp2)*ZM + RR1)*DT));

    if j=1 then do;
      max_det_add=det_add;
  max_index = 1;
end;
else do;
  if det_add > max_det_add then do;
    max_det_add = det_add;
max_index = j ;
```

```
  end;
end;
  end;

  x= cand_x[max_index,];
  d= cand_d[max_index,];
  design_x = design_x // x ;
  design_d = design_d // d;
  max_det = 1/(det(DT‘*ginv(ZM‘*(design_x‘*design_x)*inv(i(p) +
        aa*design_x‘*design_x + bb*JM*design_x‘*design_x)*ZM +RR1)*DT));

  print "Design added: ";
  print x, d;
  print "max determinant: ";
  print max_det;

end;

print "Design before exchange algorithm";
print design_x, design_d;

%*************************************************************;
%** apply exchange algorithm                                 **;
%*************************************************************;

do i=1 to max_iteration;

  %****************************************;
  %*  add one design point                **;
  %****************************************;

  do j=1 to cand_n;
    x=cand_x[j,];

den_temp1 = design_x‘*design_x + x‘*x;
den_temp2 = i(p)+ aa*design_x‘*design_x + bb*JM*design_x‘*design_x
                    + aa*x‘*x +bb*JM*x‘*x;
    det_add = 1/(det(DT‘*ginv(ZM‘*den_temp1*inv(den_temp2)*ZM + RR1)*DT));

    if j=1 then do;
      max_det_add=det_add;
  max_index = 1;
end;
else do;
  if det_add > max_det_add then do;
    max_det_add = det_add;
```

```
max_index = j ;
  end;
end;
  end;

  max_index_add = max_index;

  x_add = cand_x[max_index,];
  d_add = cand_d[max_index,];
  design_x1 = design_x // x_add ;
  design_d1 = design_d // d_add;

  %************************************;
  %** delete one design point        **;
  %************************************;

  design_x1_n = nrow(design_x1);

  do j=1 to design_x1_n;
    x=design_x1[j,];

den_temp3 = design_x'*design_x - x'*x;
den_temp4 = i(p)+ aa*design_x'*design_x + bb*JM*design_x'*design_x
                    - aa*x'*x - bb*JM*x'*x;
   det_del = 1/(det(DT'*ginv(ZM'*den_temp3*inv(den_temp4)*ZM + RR1)*DT));

    if j=1 then do;
      max_det_del=det_del;
  max_index = 1;
end;
else do;
  if det_del > max_det_del then do;
    max_det_del = det_del;
max_index = j ;
  end;
end;
  end;

  max_index_del = max_index;

  x_del= design_x1[max_index,];
  d_del= design_d1[max_index,];

  if max_index_del=1 then do;
     do k=2 to nrow(design_x1);
    if k=2 then do;
```

```
          design_x2=design_x1[2,];
  design_d2=design_d1[2,];
end;
else do;
  design_x2=design_x2 // design_x1[k,];
  design_d2=design_d2 // design_d1[k,];
end;
 end;
  end;
  else do;
     do k=1 to nrow(design_x1);
    if k=1 then do;
          design_x2=design_x1[1,];
  design_d2=design_d1[1,];
end;
else do;
  if k ^= max_index_del then do;
    design_x2=design_x2 // design_x1[k,];
    design_d2=design_d2 // design_d1[k,];
  end;
end;
 end;
  end;

  max_det2 = 1/(det(DT`*ginv(ZM`*(design_x2`*design_x2)*inv(i(p) +
       aa*design_x2`*design_x2 + bb*JM*design_x2`*design_x2)*ZM +RR1)*DT));
  print "number of interations:";
  print i ;
  print "Design to be added: ";
  print x_add, d_add;
  print "Design to be deleted: ";
  print x_del, d_del;
  print "new max determinant: ";
  print max_det2;

  if max_det2 > max_det then do;
    max_det=max_det2;
design_x=design_x2;
design_d=design_d2;
  end;
  else do;
    print "Exchange does not improve criterion.";
    print "Final design:";
print design_x, design_d ;
print "Criterion Max:";
print max_det;
```

```
print "xpx";
xpx=design_x‘*design_x;
print xpx;
    stop;
  end;
end;

if i=max_iteration+1 then do;
  print "Maximum number of iterations is reached";
  print "Final design:";
  print design_x, design_d;
  print "Criterion Max:";
  print max_det;
  xpx=design_x‘*design_x;
  print "xpx";
  print xpx;
end;
```