

Supplementary Material

This section provides the MATLAB codes for calculating low-order factor aliasing.

The MATLAB codes for generating a design matrix from a saturated design.

```
1 function [A, F] =generateIndependentMatrix(n, m, D)
2     q = n - m;
3     Hq = generate_Hq(q);
4     if any(D > size(Hq, 2))
5         error('One or more indices in D exceed the number of columns in H_q.');
```

```
6     end
7     A = zeros(n, length(D));
8     for i = 1:length(D)
9         A(1:q, i) = Hq(:, D(i));
10    end
11    F = zeros(n, m);
12    for j = 1:m
13        col_idx = n - m + j;
14        if col_idx <= length(D)
15            F(:, j) = A(:, col_idx);
16            F(col_idx, j) = F(col_idx, j) + 2;
17        else
18            error('Confounded column index exceeds the number of columns in A');
```

```
19        end
20    end
21 end
22
23
24 function Hq =generate_Hq(q)
25     H1 = zeros(q, 1);
26     H1(1) = 1;
27     Hq = H1;
28
29     for i = 2:q
30         [numRows, ~] = size(Hq);
31         Q = zeros(numRows, 1);
32         Q(i) = 1;
33         Hq_temp = Hq;
34
35         Hq = [Hq_temp, Q, mod(Hq_temp + Q, 3), mod(Hq_temp + 2*Q, 3)];
36     end
37 end
```

The MATLAB codes for constructing the defining contrast subgroup matrix.

```
1 function G =generateEffectMatrix(F)
2     [n, m] = size(F);
3     Gm = F;
4
5     uniqueCols = containers.Map();
6
7
8     for i = 1:m
9         colStr = mat2str(F(:, i)');
10        uniqueCols(colStr) = true;
11    end
12
13    for k = 2:m
```

```

14     combs = nchoosek(1:m, k);
15     numCombs = size(combs, 1);
16     factorCombos = dec2bin(0:(2^k - 1)) - '0' + 1;
17
18     for i = 1:numCombs
19         for factorCombo = factorCombos'
20             baseVec = zeros(n, 1);
21             for j = 1:k
22                 factor = factorCombo(j);
23                 baseVec = mod(baseVec + factor * F(:, combs(i, j)), 3);
24             end
25             if sum(baseVec ~= 0) <= 4
26                 colStr = mat2str(baseVec');
27                 if ~isKey(uniqueCols, colStr)
28                     Gm = [Gm, baseVec];
29                     uniqueCols(colStr) = true;
30                 end
31             end
32         end
33     end
34 end
35
36 Gmm = [];
37 for i = 1:size(Gm, 2)
38     col = Gm(:, i);
39     modCol = mod(2 * col, 3);
40     if sum(modCol ~= 0) <= 4
41         colStr = mat2str(modCol');
42         if ~isKey(uniqueCols, colStr)
43             Gmm = [Gmm, modCol];
44             uniqueCols(colStr) = true;
45         end
46     end
47 end
48
49 G = [Gm, Gmm];
50 validCols = [];
51 for i = 1:size(G, 2)
52     if sum(G(:, i) ~= 0) <= 4
53         validCols = [validCols, G(:, i)];
54     end
55 end
56
57 G = validCols;
58 end

```

The MATLAB codes for calculating the low-order factor aliasing values of the design.

```

1 function A21 = calculateA21(n, m, D)
2     G = mainFunction(n, m, D);
3     twoFIC = TwoFIC(n);
4     k_counts = zeros(1, n + 1);
5     In = eye(n);
6
7
8     accumulatedValue = 0;

```

```

9   for i =1:n
10      ei = In(:, i);
11      Mi = mod3HadamardProduct(ei, G);
12
13      k = countMatchingColumns(Mi, twoFIC);
14      k_counts(k +1) =k_counts(k +1) +1;
15
16      accumulatedValue =accumulatedValue +1;
17
18      if accumulatedValue ==n
19          break;
20      end
21  end
22
23  k_values =0:(find(k_counts, 1, 'last') -1);
24  k_counts =k_counts(1:length(k_values));
25
26  A21 = [k_values; k_counts];
27
28  disp('k values:');
29  disp(A21(1, :));
30  disp('A21(k) values:');
31  disp(A21(2, :));
32 end
33
34 function vectors =TwoFIC(n)
35     .....(The same as above, no further elaboration)
36 end
37
38 function C =mod3HadamardProduct(e, G)
39     if size(e, 1) ~= size(G, 1)
40         error('Vector e and matrix G must have the same number of rows');
41     end
42
43     C = zeros(size(e, 1), size(G, 2));
44
45     for j =1:size(G, 2)
46         C(:, j) =mod(e +G(:, j), 3);
47     end
48 end
49
50 function k =countMatchingColumns(M, twoFIC)
51     k = 0;
52     for i =1:size(M, 2)
53         for j =1:size(twoFIC, 2)
54             if isequal(M(:, i), twoFIC(:, j))
55                 k = k + 1;
56                 break;
57             end
58         end
59     end
60 end
61
62 function G =mainFunction(n, m, D)
63     .....(The same as above, no further elaboration)
64 end
65
66 function Hq =generate_Hq(q)

```

```

67 .....(The same as above, no further elaboration)
68 end
69
70 function G =generateEffectMatrix(F)
71 .....(The same as above, no further elaboration)
72 end

```

```

1 function A22 =calculateA22(n, m, D)
2     G = mainFunction(n, m, D);
3     B = generateTwoFICMatrix(n);
4     B_split =splitMatrix(B, n);
5     twoFIC =TwoFIC(n);
6     k_counts =zeros(1, nchoosek(n, 2) +1);
7     totalPairs =nchoosek(n, 2);
8     accumulatedValue =0;
9     for i =1:size(B_split, 2)
10         Bi = B_split{i};
11         Mi =mod3HadamardProduct(Bi, G);
12         k = countMatchingColumns(Mi, twoFIC);
13         k_counts(k +1) =k_counts(k +1) +1;
14         accumulatedValue =accumulatedValue +k_counts(k +1);
15         if accumulatedValue ==totalPairs
16             break;
17         end
18     end
19     k_values =0:totalPairs;
20     k_counts =k_counts(1:length(k_values));
21     zeroIndex =find(k_counts ==0, 1, 'last');
22     k_values =k_values(1:zeroIndex);
23     k_counts =k_counts(1:zeroIndex);
24
25     A22 = [k_values; k_counts];
26
27     disp('k values:');
28     disp(A22(1, :));
29     disp('A22(k) values:');
30     disp(A22(2, :));
31 end
32
33 function vectors =TwoFIC(n)
34     totalColumns =3 *nchoosek(n, 2);
35     vectors =zeros(n, totalColumns);
36     colIdx =1;
37     for i =1:n
38         for j =i+1:n
39             vectors(i, colIdx) =1;
40             vectors(j, colIdx) =1;
41             colIdx =colIdx +1;
42             vectors(i, colIdx) =1;
43             vectors(j, colIdx) =2;
44             colIdx =colIdx +1;
45             vectors(i, colIdx) =2;
46             vectors(j, colIdx) =1;
47             colIdx =colIdx +1;
48             vectors(i, colIdx) =2;
49             vectors(j, colIdx) =2;
50             colIdx =colIdx +1;

```

```

51     end
52 end
53 end
54
55 function B =generateTwoFICMatrix(n)
56     totalColumns =2 *nchoosek(n, 2);
57     B = zeros(n, totalColumns);
58     colIdx =1;
59     for i =1:n
60         for j =i+1:n
61             B(i, colIdx) =1;
62             B(j, colIdx) =1;
63             colIdx =colIdx +1;
64             B(i, colIdx) =1;
65             B(j, colIdx) =2;
66             colIdx =colIdx +1;
67         end
68     end
69 end
70
71 function B_split =splitMatrix(B, n)
72     numPairs =nchoosek(n, 2);
73     B_split =cell(1, numPairs);
74     for i =1:numPairs
75         B_split{i} =B(:, (2*i-1):(2*i));
76     end
77 end
78
79 function C =mod3HadamardProduct(A, B)
80     [nA, m] =size(A);
81     [nB, n] =size(B);
82     if nA ~= nB
83         error('Matrices A and B must have the same number of rows');
84     end
85     C = zeros(nA, m *n);
86     count =1;
87     for i =1:m
88         for j =1:n
89             C(:, count) =mod(A(:, i) +B(:, j), 3);
90             count =count +1;
91         end
92     end
93 end
94
95 function k =countMatchingColumns(M, twoFIC)
96     k = 0;
97     for i =1:size(M, 2)
98         for j =1:size(twoFIC, 2)
99             if isequal(M(:, i), twoFIC(:, j))
100                 k = k + 1;
101                 break;
102             end
103         end
104     end
105 end
106
107 function G =mainFunction(n, m, D)
108     [A, F] =generateIndependentMatrix(n, m, D);

```

```

109     G = generateEffectMatrix(F);
110 end
111
112 function [A, F] =generateIndependentMatrix(n, m, D)
113 .....(The same as above, no further elaboration)
114 end
115
116 function Hq =generate_Hq(q)
117 .....(The same as above, no further elaboration)
118 end
119
120 function G =generateEffectMatrix(F)
121 .....(The same as above, no further elaboration)
122 end

```