```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BILEVEL OPTIMIZER MAIN SCRIPT
% FOR FALL 2017 ME 6101 FINAL PROJECT ON GREEN MODULAR DESIGN OPTIMIZATION
% ENGINEERS: JAMES S COLLINS
%            BEN DUSSALT
%            NAMKHA NORSANG
%            NISHANTH KATHIRVEL
%
% PROJECT: ME 6101 GREEN MODULAR DESIGN GROUP PROJECT
% DATE: NOVEMBER 2017
% LOCATION: GEORGIA INSTITUTE OF TECHNOLOGY. ATL, GA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
clc

load Vg.mat;
load Vh.mat;
load Vs.mat;


Fstar=0.01;
fstar=0.006;

components=37;
ULiterations=2000;
LLiterations=2000;
population=700;

m=round(sqrt(components)); %inital max number of modules

% Store the alpha, beta (>=0) and the corresponding chromosome
A=cell(1,5);

ULmembers = randi(m,[population components]);
LLmembers = zeros(population, components);
ULscores = randi(10,[population 1]);

for i=1:ULiterations
    % Perform CGA on ULmembers
    ULmembers=GreenModGeneticAlgorithmUL(ULmembers,ULscores);
    % now we have a new population of ULCs
    for j=1:population
        [X,binX,~,m] = chromoSort(ULmembers(j,:)); %decompose each ULC
        F=fitnessFunctionF(vh,vs,ULmembers(j,:));
        ULscores(j,1)=F;
        alpha=(F-Fstar)/(1-Fstar);
        %%%%Evaluate Alpha(X)%%%
        if alpha<0 || isnan(alpha)
            break
        else
            binY=binX;
            Y=X;
            f=fitnessFunctionff(vg,Y);
            beta=(f-fstar)/(1-fstar);
```

```
                %%%%Evaluate Beta(Y)%%%
                if beta>=0 && ~isnan(beta)
                    [currentAPop,~]=size(A);
                    A(currentAPop+1,:)={[alpha],[beta],[F],[f],ULmembers(j,:)};
                else

                    for k=1:LLiterations
                        %%%%%perform CGA on Y%%%%%
                        Y=GreenModGeneticAlgorithmLL(Y);
                        for l=1:size(Y,1)
                            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                            f = fitnessFunctionff(vg,Y);
                            %%%%evaluate beta(Y) %%%%
                            beta=(f-fstar)/(1-fstar);
                            if beta<0 || isnan(beta)
                                break
                            else
                                Y=sum(Y,1);
                                F=fitnessFunctionF(vh,vs,Y);
                                %%%%evaluate alpha(X)%%%%%%
                                alpha=(F-Fstar)/(1-Fstar);
                                if alpha<0 || isnan(alpha)
                                    break
                                else
                                    [currentAPop,~]=size(A);
                                    A(currentAPop+1,:)={[alpha],[beta],[F],[f],Y(l,:)};
                                end
                            end
                        end
                    end
                end
            end
        end
end
```

```
function [chromosomeArr,binChromosomeArr,mk,m] = chromoSort(chromosome)
% CHROMOSOME OUTPUT FORMATTER FOR OUTPUT OF GENETIC ALGORITHM TO BE
% PROCESSED BY THE FITNESS FUNCTION F(X).
% THE FUNCTION IS USED IN THE FALL 2017 ME 6101 FINAL GROUP PROJECT ON
% GREEN MODULAR DESIGN OPTIMIZATION
%
% NOTE: TO OBTAIN THE FORMATTED CHROMOSOME USED IN F(X), USE chromoSort.m
%       INSTEAD!
%
%
% INPUT: [1XN DOUBLE] CHROMOSOME OUTPUT FROM GA
%                     EACH INDICE IN THE VECTOR IS A GENE INDICATING THE
%                     NUMBER OF MODULES TO BE GROUPED. THE INDEX OF THE
%                     GENE POSITION INDICATES THE NUMBER OF COMPONENTS IN
%                     THE MODULE
%
% OUTPUT: [MxN DOUBLE]  REFORMATTED ARRAY BREAKING THE NUMBER OF
%                       MODULES TO BE GROUPED INTO DIFFERENT ROWS.
%                       THE INDICES MARK THE POSITIONS WHERE THE MODULE
%                       NUMBERS ARE PRESENT - IE THE NUMBER OF COMPONENTS
%                       IN THE MODULES
%         [MxN DOUBLE]  REFORMATTED ARRAY SIMILAR TO chromosomeArr BUT
%                       CONSISTING OF BINARY ELEMENTS
%         [1x1 DOUBLE]  THE MAXIMUM NUMBER OF DIFFERENT ELEMENTS IN THE
%                       INPUT CHROMOSOME VECTOR
%         [1x1 DOUBLE]  THE MAXIMUM ELEMENT VALUE CONTAINED IN THE
%                       INPUT CHROMOSOME VECTOR

% ENGINEERS: JAMES S COLLINS
%            BEN DUSSALT
%            NAMKHA NORSANG
%            NISHANTH KATHIRVEL
%
% PROJECT: ME 6101 GREEN MODULAR DESIGN GROUP PROJECT
% DATE: NOVEMBER 2017
% LOCATION: GEORGIA INSTITUTE OF TECHNOLOGY. ATL, GA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    L=length(chromosome); % GET THE LENGTH OF THE CHROMOSOME INPUT IE THE
                          % TOTAL NUMBER OF COMPONENTS

    % IDENTIFING MAXIMUM MODULE NUMBER IN CHROMOSOME AND DECLARING THE
    % OUTPUT LOGICAL ARRAY
    m=max(chromosome);
    binChromosomeArr=false(L,L);
    chromosomeArr=zeros(L,L);
    % FOR LOOP TO SEPAREATE GENES INTO INDIVIDUAL ROWS MARKING DIFFERENT
    % COLUMNS
    for i=1:L
        % GENERATING LOGICAL MASK ROWS
        binChromosomeArr(i,:)=chromosome==i;
        % PORTION TO ASSIGN THE DOUBLE ARRAY THE SEPARATE VALUES
        rw=zeros(1,L);
        rw(binChromosomeArr(i,:))=i;
        chromosomeArr(i,:)=rw;
    end
    % CREATION OF LOGICAL MASK IDENTIFING
    % ROWS CONTAINING ALL FALSE INDICES

    %deleteMask=~any(binChromosomeArr,2);

    % USE THE LOGICAL MASK TO IDENTIFY AND ISOLATE ALL THE ROWS CONTAINING
    % TRUE ELEMENTS IN THE COLUMNS OR DOUBLES ~= 0


    %%binChromosomeArr(deleteMask,:)=[];

    %chromosomeArr(deleteMask,:)=[];

    % DETERMINE THE PARAMENTER m USED IN LATER CALCULATIONS AS THE NUMBER
    % OF FILLED ROWS IN THE CHROMOSOME ARRAY
    [mk,~]=size(chromosomeArr);
    % CONVERT THE LOGICAL TO TYPE DOUBLE FOR FITNESS FUNCTION CALCULATION
    binChromosomeArr=chromosomeArr~=0;
    binChromosomeArr=double(binChromosomeArr);

end
```

Published with MATLAB® R2017b

```matlab
function [multiDimArray,mk] = chromoSortMultiDim(chromeArray)
% CHROMOSOME OUTPUT FORMATTER FOR OUTPUT OF GENETIC ALGORITHM TO BE
% PROCESSED BY THE FITNESS FUNCTION f(X).
% THE FUNCTION IS USED IN THE FALL 2017 ME 6101 FINAL GROUP PROJECT ON
% GREEN MODULAR DESIGN OPTIMIZATION
%
% NOTE: TO OBTAIN THE FORMATTED CHROMOSOME USED IN F(X), USE chromoSort.m
%       INSTEAD!
%
% INPUT: [MXN DOUBLE]       CHROMOSOME OUTPUT FROM GA
%                           EACH INDICE IN THE VECTOR IS A GENE INDICATING
%                           THE NUMBER OF MODULES TO BE GROUPED. THE INDEX
%                           OF THE GENE POSITION INDICATES THE NUMBER OF
%                           COMPONENTS
%
% OUTPUT: [MxMxN DOUBLE]    REFORMATTED LOGICAL ARRAY BREAKING THE NUMBER
%                           OF MODULES TO BE GROUPED INTO DIFFERENT ROWS.
%                           THE TRUE/ONE INDICES MARK THE POSITIONS WHERE
%                           THE MODULES ARE PRESENT - IE THE NUMBER OF
%                           COMPONENTS IN THE MODULE
%         [1x1 DOUBLE]      VALUE OF THE TOTAL NUMBER OF DIFFERENT
%                           ELEMENTS CONTAINED IN THE DIFFERENT
%                           CHROMOSOMES. THIS CORRESPONDS TO THE NUMBER OF
%                           LOWER LEVEL CHROMOSOMES THAT HAVE BEEN
%                           DECOMPOSED
%
% ENGINEERS: JAMES S COLLINS
%            BEN DUSSALT
%            NAMKHA NORSANG
%            NISHANTH KATHIRVEL
%
% PROJECT: ME 6101 GREEN MODULAR DESIGN GROUP PROJECT
% DATE: NOVEMBER 2017
% LOCATION: GEORGIA INSTITUTE OF TECHNOLOGY. ATL, GA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INDEX THE NUMBER OF ROWS AND COLUMNS OF THE INPUT CHROMOSOME ARRAY
% n IS THE NUMBER OF COMPONENTS
[R,n]=size(chromeArray);
% INITIALIZE THE OUTPUT AS A R3 ARRAY
multiDimArray=zeros(1,1,n);
mk=0;
% LOOP THROUGH THE ROWS OF THE INPUT ARRAY
for i=1:1:R
    % CALL FORMATTING FUNCTION FOR EACH 1xN CHROMOSOME VECTOR
    % THE OUTPUT IS A MxN ARRAY
    [~,binArr,mki,m]=chromoSort(chromeArray(i,:));
    % INSERT THE R2 ARRAY INTO A THIRD DIM LAYER OF THE R3 ARRAY. EACH OF
    % THESE THIRD DIM SLICES CORRESPONDS TO A ROW IN THE FUNCTION'S INPUT
    % CHROMOSOME ARRAY

    multiDimArray(i,1:mk,:)=binArr;
    mk=mk+mki;
end
```

```
function FX = fitnessFunctionF(vh,vs,chromosome)
% THIS FUNCTION CALCULATES THE FITNESS FUNCTION F(X) DESIGN
% MEASURE. THIS FUNCTION SHOULD BE USED THROUGH THE BILEVEL OPTIMIZATION
% ALONG WITH THE SECOND FITNESS FUNCTION f(X)
% FOR THE FORMULA, SEE EQUATION 8
%
% INPUT:  [NxN DOUBLE]  CCF ARRAY DEFINED AS v^s IN THE EQUATION
%         [NxN DOUBLE]  CCF ARRAY DEFINED AS v^h IN THE EQUATION
%         [1xN DOUBLE]  DESIGN CHROMOSOME VECTOR
% OUTPUT: [1x1 DOUBLE]  FIRST NUMERATOR TERM CALCULATED FOR THE F(X)
%                       DESIGN MEASUREMENT
%
% ENGINEERS: JAMES S COLLINS
%            BEN DUSSALT
%            NAMKHA NORSANG
%            NISHANTH KATHIRVEL
%
% PROJECT: ME 6101 GREEN MODULAR DESIGN GROUP PROJECT
% DATE: NOVEMBER 2017
% LOCATION: GEORGIA INSTITUTE OF TECHNOLOGY. ATL, GA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 [~,binChromosomeArray,~,m]=chromoSort(chromosome);

 [FNT1,m]=FNumerator1(vh,vs,m,binChromosomeArray);
 [FNT2,~]=FNumerator2(vh,vs,m,binChromosomeArray);

 FX=(FNT1-FNT2)/m;




end


function [FNumeratorTerm1,numtest1] = FNumerator1(vh,vs,m,chromArray)
% THIS FUNCTION CALCULATES THE FIRST NUMERATOR TERM OF THE F(X) DESIGN
% MEASURE. THIS FUNCTION ALONG WITH FNumerator2 SHOULD BE USED THROUGH THE
% BILEVEL OPTIMIZATION
% FOR THE FORMULA, SEE EQUATION 8
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SETTING THE COMPONENT NUMBER
n=37;
% DECLARING ITERATION VARIBLES
FNumeratorTerm1=0;
numo=0;
denom=0;
numo2=0;

numtest1=[];
% STARTING FIRST, EXTERIOR SUMMATION FOR LOOP
    for k=1:1:m
        chkRow = sum(chromArray(k,:));
        if chkRow~=0

            % TWO NESTED SUMMATION THAT MAKE UP THE NUMERATOR FOR THE TERM
                for i=1:1:n
                    for j=1:1:n
                        % PRODUCT OF INDEXED VALUES ON THE NUMERATOR
                        % INCLUDED IN THE DOUBLE SUMMATION
                        v1=vh(i,j);
                        v2=vs(i,j);
                        x1=chromArray(k,i);
                        x2=chromArray(k,j);
                    end
                end
                numo = numo2;
                numo2 = 0;
                % THIS BLOCK DEFINES THE SQUARED
                % SUMMATION TERM IN THE DENOMINATOR
                for l=1:1:n
                    denom=denom+chromArray(k,l);
                end
                denom=(denom)^2;
                % EXTERNAL SUMMATION
                FNumeratorTerm1=FNumeratorTerm1+(numo/denom);
                denom = 0;
        end
    end
end

function [FNumeratorTerm2,numotest2] = FNumerator2(vh,vs,m,chromArray)
% THIS FUNCTION CALCULATES THE SECOND NUMERATOR TERM OF THE F(X) DESIGN
% MEASURE. THIS FUNCTION ALONG WITH FNumerator2 SHOULD BE USED THROUGH THE
% BILEVEL OPTIMIZATION
% FOR THE FORMULA, SEE EQUATION 8
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

n=37; % SETTING THE COMPONENT NUMBER
FNumeratorTerm2=0; % DECLARING ITERATION VARIBLES
numo2=0;
denom=0;

numotest2=[];
% STARTING FIRST, EXTERIOR SUMMATION FOR LOOP
    for k=1:1:m
        chkRow = sum(chromArray(k,:));
        if chkRow~=0

        % TWO NESTED SUMMATION THAT MAKE UP THE NUMERATOR FOR THE TERM
            for i=1:1:n
                for j=1:1:n
                    % PRODUCT OF INDEXED VALUES ON THE NUMERATOR
                    % INCLUDED IN THE DOUBLE SUMMATION
                    v1=vh(i,j);
                    v2=vs(i,j);
                    x1=chromArray(k,i);
                    x2=abs(1-(chromArray(k,j)));
                end
            end
            numo = numo2;
            numo2 = 0;
            % THIS BLOCK DEFINES THE SQUARED
            % SUMMATION TERM IN THE DENOMINATOR
            for l=1:1:n
                denom=denom+chromArray(k,l);
            end
            denom=(n-denom)^2;
            % EXTERNAL SUMMATION
            FNumeratorTerm2=FNumeratorTerm2+(numo/denom);
            denom = 0;
        end
    end
end
```

```
function fY = fitnessFunctionff(vg,crossOverChromosome)
% THIS FUNCTION CALCULATES THE FITNESS FUNCTION f(Y) DESIGN
% MEASURE. THIS FUNCTION SHOULD BE USED THROUGH THE BILEVEL OPTIMIZATION
% ALONG WITH THE FIRST FITNESS FUNCTION F(X)
% FOR THE FORMULA, SEE EQUATION 9
%
% INPUT:  [NxN DOUBLE]  CCF ARRAY DEFINED AS v^g IN THE EQUATION
%         [MxN DOUBLE]  ARRAY OF LLCS FOR EACH ULC MODULE
%
% OUTPUT: [1x1 DOUBLE]  FITNESS f(Y) DESIGN MEASURE
%
% ENGINEERS: JAMES S COLLINS
%            BEN DUSSALT
%            NAMKHA NORSANG
%            NISHANTH KATHIRVEL
%
% PROJECT: ME 6101 GREEN MODULAR DESIGN GROUP PROJECT
% DATE: NOVEMBER 2017
% LOCATION: GEORGIA INSTITUTE OF TECHNOLOGY. ATL, GA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 [FNT1,mk] = ffNumerator1(vg,crossOverChromosome);
 [FNT2,~] = ffNumerator2(vg,crossOverChromosome);
 m = max(max(crossOverChromosome));

 denom = 0;
 for i=1:1:m
   denom = denom + mk;
 end

 fY=(FNT1-FNT2)/denom;

end

function [ffNumeratorTerm1,mk] = ffNumerator1(vg,crossOverChromosome)
% THIS FUNCTION CALCULATES THE FIRST NUMERATOR TERM OF THE f(Y) DESIGN
% MEASURE. THIS FUNCTION ALONG WITH ffNumerator2 SHOULD BE USED THROUGH THE
% BILEVEL OPTIMIZATION
% FOR THE FORMULA, SEE EQUATION 9
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SETTING THE COMPONENT NUMBER
n=22;
% DECLARING ITERATION VARIABLES
ffNumeratorTerm1=0;
numo=0;
denom=0;
% SORT THE INPUT INTO A R3 BINARY ARRAY
[binChromArray,mk] = chromoSortMultiDim(crossOverChromosome);
% ISOLATE THE LARGEST ELEMENT VALUE AS m
[ym,~] = size(crossOverChromosome);

% STARTING FIRST, EXTERIOR SUMMATION FOR LOOP
    for k = 1:1:ym
```

```
        [~,~,mk,~] = chromoSort(crossOverChromosome(k,:));
        for j=1:1:mk
            chkVec = sum(binChromArray(k,j,:));
            if chkVec~=0

            % TWO NESTED SUMMATION THAT MAKE UP THE NUMERATOR FOR THE TERM
                for i=1:1:n
                    for l=1:1:n
                        % PRODUCT OF INDEXED VALUES ON THE NUMERATOR
                        % INCLUDED IN THE DOUBLE SUMMATION
                        v1=vg(i,l);
                        y1=binChromArray(k,j,i);
                        y2=binChromArray(k,j,l);
                        numo=numo+(y1*y2*v1);
                    end
                end

                % THIS BLOCK DEFINES THE SQUARED
                % SUMMATION TERM IN THE DENOMINATOR
                for l=1:1:n
                    denom=denom+binChromArray(k,j,l);
                end
                denom=(denom)^2;
                % EXTERNAL SUMMATION
                ffNumeratorTerm1=ffNumeratorTerm1+(numo/denom);
                numo=0;
                denom=0;
            end
        end
    end
end

function [ffNumeratorTerm2,mk] = ffNumerator2(vg,crossOverChromosome)
% THIS FUNCTION CALCULATES THE FIRST NUMERATOR TERM OF THE f(Y) DESIGN
% MEASURE. THIS FUNCTION ALONG WITH ffNumerator2 SHOULD BE USED THROUGH THE
% BILEVEL OPTIMIZATION
% FOR THE FORMULA, SEE EQUATION 9
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



% SETTING THE COMPONENT NUMBER
n=22;
% DECLARING ITERATION VARIBLES
ffNumeratorTerm2=0;
numo=0;
denom=0;

% SORT THE INPUT INTO A R3 BINARY ARRAY
[binChromArray,mk] = chromoSortMultiDim(crossOverChromosome);
% ISOLATE THE LARGEST ELEMENT VALUE AS m
[ym,~] = size(crossOverChromosome);

% STARTING FIRST, EXTERIOR SUMMATION FOR LOOP
    for k = 1:1:ym
```

```
        for j=1:1:mk
            chkVec = sum(binChromArray(k,j,:));
            if chkVec~=0
            % TWO NESTED SUMMATION THAT MAKE UP THE NUMERATOR FOR THE TERM
                for i=1:1:n
                    for l=1:1:n
                        % PRODUCT OF INDEXED VALUES ON THE NUMERATOR
                        % INCLUDED IN THE DOUBLE SUMMATION
                        v1=vg(i,l);
                        y1=binChromArray(k,j,i);
                        y2=abs(1-binChromArray(k,j,l));
                        numo=numo+(y1*y2*v1);
                    end
                end
                % THIS BLOCK DEFINES THE SQUARED
                % SUMMATION TERM IN THE DENOMINATOR
                for l=1:1:n
                    denom=denom+binChromArray(k,j,l);
                end
                denom=(n-denom)^2;
                % EXTERNAL SUMMATION
                ffNumeratorTerm2=ffNumeratorTerm2+(numo/denom);
                denom=0;
                numo=0;
            end
        end
    end
end
```

```
function children  = GreenModCrossoverScattered(parents,mateIndex)
%   EXECUTES A POSITION INDEPENDENT CROSSOVER OF CHROMOSOME ELEMENTS
%   M ROWS OF DESIGN CHROMOSOMES ARE MATCHED FOR CROSSOVER BY A SELECTION
%   INDEX FROM THE GreenModSelection.m FUNCTION
%
%   [MxN DOUBLE] = GreenModCrossoverScattered([MxN DOUBLE],[Mx1 DOUBLE])
%
%   INPUTS:
%       parents : ARRAY CONSISTING OF M DESIGN CHROMOSOME VECTORS FOR N
%                 DESIGN COMPONENTS
%       mateIndex :      VECTOR IDENTIFYING THE ROWS IN THE PARENT ARRAY
%                        TO BE BRED WITH THE MATCHING ROW IN THE ARRAY
%                        WHERE THE VECTOR'S ELEMENT IS POSITIONED.
%                        IE. A FIRST ELEMENT IN selectionIndex of 3 MEANS
%                        THAT THE THIRD ROW IN parents IS TO BE MATED WITH
%                        THE FIRST ROW IN parents TO PRODUCE THE FIRST ROW
%                        OF THE CHILDREN OUTPUT ARRAY
%
%   OUTPUTS:
%       children : NEW CHROMOSOME ARRAY AFTER CROSSOVER COMPLETION
%
%   ENGINEERS: JAMES COLLINS & NAMKHA NORSANG
%   PROJECT: ME 6101 GREEN MODULAR DESIGN GROUP PROJECT
%   DATE: NOVEMBER 2017
%   LOCATION: GEORGIA INSTITUTE OF TECHNOLOGY. ATL, GA
%
%   NOTE: THIS FUNCTION IS BASED ON MATLAB'S crossoverScattered.m FUNCTION

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DETERMINE SIZE OF CHROMOSOME VECTORS AND NUMBER IN POPULATION
[nchildren,GenomeLength] = size(parents);
% INITALIZE CHILDREN OUTPUT ARRAY
children = zeros(nchildren,GenomeLength);
% INITALIZED NEW ARRAY TO MATCH TO PARENT ARRAY FOR MATING
mateArray = parents(mateIndex,:);
% INITIALIZE INDEX
index = 1;
for i=1:nchildren/2
    % PAIR THE PARENTS
    r1 = parents(index,:);
    index = index+1;
    r2 = mateArray(index,:);
    % SELECT 1/2 THE GENES FROM EACH PARENT TO SWAP
    for j = 1:GenomeLength
        if(rand > 0.5)
            children(index-1,j) = r1(j);
            children(index,j) = r2(j);
        else
            children(index-1,j) = r2(j);
            children(index,j) = r1(j);
        end
    end
    index = index + 1;
end
end
```

```matlab
function expectation = GreenModFitScalingRank(scores)
%   RANK BASED FITNESS SCALING. SCALES THE Mx1 SCORES FROM THE F(X) AND
%   f(X) FITNESS FUNCTION TO READABLE A READABLE FORMAT FOR THE SELECTION
%   FUNCTION GreenModSelection.m.
%
%   [Mx1 DOUBLE] = GreenModFitScalingRank([Mx1 DOUBLE],
%
%   INPUTS:
%       scores : VECTOR OF NUMBERS CORRESPONDING TO THE RESULTS OF EACH
%       CHROMOSOME (ROW) EVALUATED BY THE FITNESS FUNCTIONS F(X) OR f(X)
%
%   OUTPUTS:
%       expectation : VECTOR OF SCALED RESULTS TO THEN BE INPUT INTO THE
%       SELECTION FUNCTION GreenModSelection.m
%
%   NOTE: IT IS ASSUMED THAT THE SCORES ARE MEANT TO BE MAXIMIZED FOR
%   FUTURE GENERATIONS - IE, ASCENDING SCORES CORRESPOND TO ASCENDING
%   LEVELS OF FITNESS.
%
%   ENGINEERS: JAMES COLLINS & NAMKHA NORSANG
%   PROJECT: ME 6101 GREEN MODULAR DESIGN GROUP PROJECT
%   DATE: NOVEMBER 2017
%   LOCATION: GEORGIA INSTITUTE OF TECHNOLOGY. ALT, GA
%
%   NOTE: THIS FUNCTION IS BASED ON MATLAB'S fitScalingrank.m FUNCTION

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


[nParents,~] = size(scores);
scores = scores(:);
[~,i] = sort(scores);

expectation = zeros(size(scores));
expectation(i) = 1 ./ ((1:length(scores)) .^ 0.5);

expectation = nParents * expectation ./ sum(expectation);
```

```
function [newGen] = GreenModGeneticAlgorithmLL(prevGen)
% GENETIC ALGORITHM TO PROCESS A GIVEN SET OF CHROMOSOMES DEFINING A
% POPULATION AND USE FITNESS SCORES TO PRODUCE A NEW GENERATION WITH
% CROSSOVER AND MUTATION ALTERATIONS.
%
%   NOTE: THIS IS USED IN THE LOWER LEVEL OPTIMIZATION!!!
%         USE GreenModGeneticAlgorithmUL.m FOR THE UPPER LEVEL CHROMOSOME
%         OPTIMIZATION
%         THE ONLY DIFFERENCE BETWEEN THIS SCRIPT AND
%         GreenModGeneticAlgorithmUL.m IS THAT IT DOES NOT TAKE INTO
%         ACCOUNT FITNESS SCORES FOR THE CROSSOVER. CROSSOVER SELECTION IS
%         THEREFORE DONE RANDOMLY
%
%   INPUTS:
%       [PxN DOUBLE] ARRAY OF DESIGN VECTORS DEFINING A POPULATION FOR A
%                    ITERATIVE GENERATION
%       [Px1 DOUBLE] SCORE OF THE INDIVIDUALS (CHROMOSOMES THAT ARE THE
%                    ROWS OF THE ARRAY) WHEN TESTED AGAINST A FITNESS
%                    FUNCITON
%   OUTPUTS:
%       [PxN DOUBLE] ARRAY OF ALTERED POPULATION THAT DEFINES THE NEXT
%                    GENERATION IN THE OPTIMIZATION
%
%
% ENGINEERS: JAMES S COLLINS
%            BEN DUSSALT
%            NAMKHA NORSANG
%            NISHANTH KATHIRVEL
%
% PROJECT: ME 6101 GREEN MODULAR DESIGN GROUP PROJECT
% DATE: NOVEMBER 2017
% LOCATION: GEORGIA INSTITUTE OF TECHNOLOGY. ATL, GA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% IDENTIFY THE NUMBER OF INDIVIDUALS IN THE POPULATION
nParents = size(prevGen,1);
% RANDOMLY DETERMINE THE SELECTION INDEX
mateIndex = randi(nParents,[nParents,1]);
% CALL THE CROSSOVER FUNCTION TO MATE DIFFERENT INDIVIDUALS IN ROWS OF THE
% POPULATION ARRAY
newGen = GreenModCrossoverScattered(prevGen,mateIndex);
% INTRODUCE MUTATIONS
newGen = GreenModMutation(newGen);
end
```

```matlab
function [newGen] = GreenModGeneticAlgorithmUL(prevGen,prevGenScores)
% GENETIC ALGORITHM TO PROCESS A GIVEN SET OF CHROMOSOMES DEFINING A
% POPULATION AND USE FITNESS SCORES TO PRODUCE A NEW GENERATION WITH
% CROSSOVER AND MUTATION ALTERATIONS.
%
%   NOTE: THIS SHOULD BE USED WITH THE UPPER LEVEL OPTIMIZATION!
%         UNLIKE GreenModGeneticAlgorithmLL.m THIS CODE CONSIDERES THE
%         FITNESS PERFORMANCE OF INDIVIDUALES BY THE SECOND INPUT OF
%         SCORES (FROM THE F(X) FITNESS FUNCTION)
%
%   INPUTS:
%       [PxN DOUBLE] ARRAY OF DESIGN VECTORS DEFINING A POPULATION FOR A
%                    ITERATIVE GENERATION
%       [Px1 DOUBLE] SCORE OF THE INDIVIDUALS (CHROMOSOMES THAT ARE THE
%                    ROWS OF THE ARRAY) WHEN TESTED AGAINST A FITNESS
%                    FUNCITON
%   OUTPUTS:
%       [PxN DOUBLE] ARRAY OF ALTERED POPULATION THAT DEFINES THE NEXT
%                    GENERATION IN THE OPTIMIZATION
%
%
% ENGINEERS: JAMES S COLLINS
%            BEN DUSSALT
%            NAMKHA NORSANG
%            NISHANTH KATHIRVEL
%
% PROJECT: ME 6101 GREEN MODULAR DESIGN GROUP PROJECT
% DATE: NOVEMBER 2017
% LOCATION: GEORGIA INSTITUTE OF TECHNOLOGY. ATL, GA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SCALE THE FITNESS FUNCTION SCORES
expectation = GreenModFitScalingRank(prevGenScores);
% DERIVE THE CORRESPONDING INDEX VECTOR FOR CROSSOVER
mateIndex = GreenModSelection(expectation);
% CALL THE CROSSOVER FUNCTION TO MATE DIFFERENT INDIVIDUALS IN ROWS OF THE
% POPULATION ARRAY
newGen = GreenModCrossoverScattered(prevGen,mateIndex);
% INTRODUCE MUTATIONS
newGen = GreenModMutation(newGen);
end
```

Published with MATLAB® R2017b

```matlab
function childrenPostMutation = GreenModMutation(children)
%   INTRODUCES RANDOM MUTATION INTO THE ARRAY OF CHROMOSOMES AFTER
%   SELECTION AND CROSSOVER. MUTATED GENES ARE RANDOMLY DISTRIBUTED OVER
%   THE CHROMOSOME.
%
%   [MxN DOUBLE] = GreenModMutation([MxN DOUBLE]
%
%   INPUTS:
%       children : ARRAY OF POPULATION CHROMOSOMES AFTER SELECTION AND
%       CROSSOVER SEQUENCES HAVE BEEN IMPLEMENTED
%
%   OUTPUTS:
%       childrenPostMutation : RESULT ARRAY OF CHROMOSOME VECTORS AFTER
%       RANDOM MUTATION
%
%   ENGINEERS: JAMES COLLINS & NAMKHA NORSANG
%   PROJECT: ME 6101 GREEN MODULAR DESIGN GROUP PROJECT
%   DATE: NOVEMBER 2017
%   LOCATION: GEORGIA INSTITUTE OF TECHNOLOGY. ALT, GA
%
%   NOTE: THIS FUNCTION IS BASED ON MATLAB'S mutationuniform.m FUNCTION

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    [PopRow,chromosomeLength] = size(children);
    % SET 5% MUTATION RATE
    mutationRate = 0.05;

    % PERFORM MUTATION EVALUATION OVER ALL CHILDREN
    for i=1:PopRow
        % RANDOMLY DETERMINE THE INDICES THAT COULD BE SELECTED FOR A
        % MUTATION
        mutationPoints = find(rand(1,chromosomeLength)...
            < mutationRate);
            % LOOP THROUGH THE SELECTED MUTATION POINTS TO DETERMINE NEW
            % VALUES
            for j=mutationPoints
                % DERIVE A RANDOM NEW ELEMENT VALUE BETWEEN 1-37
                % IE BETWEEN 1 AND 37 COMPONENTS IN THE DESIGN VECTOR
                mutValue = randi(chromosomeLength);
                % IF THE RANDOMLY DETERMINED MUTATION ELEMENT VALUE IS THE
                % SAME AS THE EXISTING ELEMENT, REDO UNTIL A DIFFERENT
                % VALUE IS REACHED
                while mutValue == children(i,j)
                    mutValue = randi(chromosomeLength);
                end
                % ASSIGN NEW VALUE
                children(i,j) = mutValue;
            end
    end
    % INITIALIZE OUTPUT FROM MODFIED INPUT ARRAY
    childrenPostMutation = children;
end
```

```matlab
function parentIndex = GreenModSelection(expectation)
%   GENERATES INDEX OF PARENTS SELECTED AND PAIRED FOR BREEDING WITH THE
%   ARRAY'S ROWS THAT WAS EVALUATED BY THE FITNESS FUNCTION THAT RESULTED
%   IN THE EXPECTATION INPUT
%
%   [MxN DOUBLE] = GreenModSelection([Mx1 DOUBLE])
%
%   INPUTS:
%       expectation : SCALED RESULT FROM GreenModFitScalingRank.m FUNCTION
%
%   OUTPUTS:
%       parentIndex : INDEX OF SELECTED CHROMOSOMES FOR BREEDING. THE
%                     ORIGINAL ROWS ARE INDEXED WITH THIS OUTPUT TO
%                     GENERATE A NEW ARRAY OF CHROMOSOMES TO BE PAIRED
%                     WITH THE ORIGINAL ROWS OF CHROMOSOMES.
%                     IE. IF THE RESULT OF arr(parents) GIVES A VECTOR C
%                     IN ROW 1 IT SHOULD BE BRED WITH VECTOR A IN ROW 1
%                     OF THE ORIGINAL ARRAY PASSED INTO THE FITNESS
%                     FUNCTION F(X) OR f(X)
%
%   ENGINEERS: JAMES COLLINS & NAMKHA NORSANG
%   PROJECT: ME 6101 GREEN MODULAR DESIGN GROUP PROJECT
%   DATE: NOVEMBER 2017
%   LOCATION: GEORGIA INSTITUTE OF TECHNOLOGY. ATL, GA
%
%   NOTE: THIS FUNCTION IS BASED ON MATLAB'S selectionstocunif.m FUNCTION

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


assert(length(expectation)>1,['AN ERROR OCCURED: FITNESS/POPULATION',...
    'SIZE IS TOO SMALL']);
nParents = length(expectation);
wheel = cumsum(expectation)/nParents;
parentIndex = zeros(1,nParents);

% STEPSIZE
stepSize = 1/nParents;

% POSITION SELECTION
position = rand * stepSize;

lowest = 1;

% ASSIGNS A PARENT TO EACH POSITION OF THE ROULETTE WHEEL
for i = 1:nParents
    for j = lowest:length(wheel)
        if(position < wheel(j))
            parentIndex(i) = j;
            lowest = j;
            break;
        end
    end
    position = position + stepSize; % ITERATE TO NEXT STEP
end
```

```
parentIndex=parentIndex';
end
```