```
function [Result] = SOSDConstBoundsv3(DIST1,DIST2)
% Finds the min number where DIST1 (the comparison distribution) SOSD DIST2 (the base distribution) and
% the min number where DIST2 SOSD DIST1 in addition to a variety of
% descriptive statistics for DIST1 and DIST2

if size(DIST1,2) > 1
    error('DIST1 must be a column vector!')
end

if size(DIST2,2) > 1
    error('DIST2 must be a column vector!')
end

% Output Result row definitions
MEANDIST1        =  1;
SDDIST1          =  2;
CVDIST1          =  3;
MAXDIST1         =  4;
MINDIST1         =  5;
PRCROPFAILDIST1  =  6;
DIST1MINPROP     =  7;
MEANDIST2        =  8;
SDDIST2          =  9;
CVDIST2          = 10;
MAXDIST2         = 11;
MINDIST2         = 12;
PRCROPFAILDIST2  = 13;
DIST2MINPROP     = 14;
MEANDIFF         = 15;
SDDIFF           = 16;
CVDIFF           = 17;
CFDIFF           = 18;
RELCOMPSOSDBASE  = 19;
RELBASESOSDCOMP  = 20;
DELTARISK        = 21;

Result           = -999999 * ones(21,1); % Initialize Results

% Calculate the mean, standard deviation, coeffcient of variation, minimum
% and maximum for DIST1
Result(MEANDIST1,1) = mean(DIST1);
Result(SDDIST1,1)   = std(DIST1);
if Result(MEANDIST1,1) > 0
    Result(CVDIST1,1)   = Result(SDDIST1,1) / Result(MEANDIST1,1);
end
Result(MAXDIST1,1)  = max(DIST1);
Result(MINDIST1,1)  = min(DIST1);

ldist1              = length(DIST1);

% Calculate the mean, standard deviation, coeffcient of variation, minimum
% and maximum for DIST2
Result(MEANDIST2,1) = mean(DIST2);
Result(SDDIST2,1)   = std(DIST2);
if Result(MEANDIST2,1) > 0
    Result(CVDIST2,1)   = Result(SDDIST2,1) / Result(MEANDIST2,1);
end
Result(MAXDIST2,1)  = max(DIST2);
Result(MINDIST2,1)  = min(DIST2);

ldist2              = length(DIST2);

% Calculate differences in descriptive statistics between DIST2 and DIST1
if Result(MEANDIST1,1) == 0 && Result(MEANDIST2,1) == 0
    Result(MEANDIFF,1)  = -999999;
else
    Result(MEANDIFF,1)  = Result(MEANDIST1,1) - Result(MEANDIST2,1);
end

if Result(SDDIST1,1) == 0 && Result(SDDIST2,1) == 0
    Result(SDDIFF,1)  = -999999;
else
    Result(SDDIFF,1)  = Result(SDDIST1,1) - Result(SDDIST2,1);
end

if Result(CVDIST1,1) >= 0 && Result(CVDIST2,1) >= 0
    Result(CVDIFF,1)  = Result(CVDIST1,1) - Result(CVDIST2,1);
else
    Result(CVDIFF,1)  = -999999;
end

% Initialize Thresholds for Stopping Golden Section Search
thresh     = 0.00001;
bailthresh = 1000000;

```

```
if Result(MEANDIST1,1) > 0 || Result(MEANDIST2,1) > 0

    % Calculate Probability of crop failiure for DIST1
    CropFailDIST1                       = DIST1;
    CropFailDIST1cond                   = DIST1(:,1) ~= 0;
    CropFailDIST1(CropFailDIST1cond,:)  = [];
    Result(PRCROPFAILDIST1,1)           = length(CropFailDIST1) / ldist1;

    % Calculate Probability of crop failiure for DIST1
    CropFailDIST2                       = DIST2;
    CropFailDIST2cond                   = DIST2(:,1) ~= 0;
    CropFailDIST2(CropFailDIST2cond,:)  = [];
    Result(PRCROPFAILDIST2,1)           = length(CropFailDIST2) / ldist2;

    tcomp = Result(MAXDIST2,1) - Result(MINDIST1,1);  % Maximum amount that DIST1 can be shifted to ensure it is SOSD
    tbase = Result(MINDIST2,1) - Result(MAXDIST1,1);  % Maximum amount that DIST1 can be shifted back to ensure DIST2 is SOSD

    % Initialize Golden Section Search upper and lower starting points
    upper  = 0;
    lower  = 0;
    if tcomp > tbase
        upper  = tcomp;
        lower  = tbase;
    elseif tcomp < tbase
        upper  = tbase;
        lower  = tcomp;
    else
        upper  = tbase + 10;
        lower  = tbase - 10;
    end

    % Initialize Flags to Test for Convergence
    bail = 0;
    converge = 0;
    while converge ~= 1 && bail < bailthresh % Find the minimum proportion that makes DIST1 SOSD DIST2
        middle = (lower + upper) / 2;
        if SOSDIntegralTestv3(DIST1 + middle, DIST2) == 1
            upper = middle;
        else
            lower = middle;
        end
        if lower > upper
            error('lower > upper!')
        end
        if (upper - lower) <= thresh  % Convergence acheived when upper and lower are within thresh tolerance
            converge = 1;
        end
        bail = bail + 1;
    end

    if converge == 1
        Result(DIST1MINPROP,1)    = upper;
        if Result(MEANDIST2,1) > 0
            Result(RELCOMPSOSDBASE,1) = upper / Result(MEANDIST2,1);
        end
    else   % Golden Section Search Failed to converge because Threshold not met
        Result(DIST1MINPROP,1)    = -777777;
        Result(RELCOMPSOSDBASE,1) = -777777;
    end

    % Initialize Golden Section Search upper and lower starting points
    if tcomp > tbase
        upper  = tcomp;
        lower  = tbase;
    elseif tcomp < tbase
        upper  = tbase;
        lower  = tcomp;
    else
        upper  = tbase + 10;
        lower  = tbase - 10;
    end

    % Initialize Flags to Test for Convergence
    bail = 0;
    converge = 0;
    while converge ~= 1 && bail < bailthresh % Find the minimum proportion that makes DIST2 SOSD DIST1
        middle = (lower + upper) / 2;
        if SOSDIntegralTestv3(DIST2, DIST1 + middle) == 1
            lower = middle;
        else
            upper = middle;
        end
        if lower > upper
            error('lower > upper!')
        end
        if (upper - lower) <= thresh % Convergence acheived when upper and lower are within thresh tolerance
            converge = 1;
        end
        bail = bail + 1;
    end

    if converge == 1
        Result(DIST2MINPROP,1)    = lower;
        if Result(MEANDIST2,1) > 0
            Result(RELBASESOSDCOMP,1) = lower / Result(MEANDIST2,1);
        end
    else
        Result(DIST2MINPROP,1)    = -777777;
        Result(RELBASESOSDCOMP,1) = -777777;
    end

    if Result(PRCROPFAILDIST1,1) >= 0 && Result(PRCROPFAILDIST2,1) >= 0
        Result(CFDIFF,1)  = Result(PRCROPFAILDIST1,1) - Result(PRCROPFAILDIST2,1);
    else
        Result(CFDIFF,1)  = -999999;
    end

    % Categorize Risk: DIST1 More Risky (-1)/less Risky (1)/ Indeterminant (0) compared to DIST2
    if Result(DIST1MINPROP,1) <= -777777 || Result(DIST2MINPROP,1) <= -777777
        Result(DELTARISK,1) = -999999;
    elseif Result(DIST1MINPROP,1) > 0 && Result(DIST2MINPROP,1) > 0
        Result(DELTARISK,1) = -1;
    elseif Result(DIST1MINPROP,1) < 0 && Result(DIST2MINPROP,1) < 0
        Result(DELTARISK,1) = 1;
    else
        Result(DELTARISK,1) = 0;
    end
end
```