```matlab
%% Overview and Assumptions:
%
% Project 2
% Group 3
%
% Assumptions:
%
%   (1) Catastrophes impact annual reproduction rate, not annual survival
%       rate
%   (2) x0 is the same for Problems 3 and 5 (best-case, worst-case, and
%       catastrophe-case)
%   (3) Each time step is 1 year.
%
% Notes:
%
%   (1) x is the population distribution after time = 0 years. The first
%       entry in x is time = 1 year. The initial population is a separate
%       variable, x0.

%% Inputs (Vectors Used to Build Leslie Matrices):

clear all; close all; clc

S_worst = [0.32 0.68.*ones(1,15)]; % Survival rate (worst-case)
S_best = [0.34 0.71.*ones(1,15)]; % Survival rate (best-case)

R_worst = [0.60 0.60 1.15.*ones(1,14)]; % Reproduction rate (worst-case)
R_best = [0.63 0.63 1.20.*ones(1,14)]; % Reproduction rate (best-case)

%% Problem 2: Long-Term Growth Rate and Distribution (Best-Case)

% Build Leslie matrix:
Lbest = diag(S_best(1:end-1),-1);
Lbest(end,end) = S_best(end);

Lbest(1,:) = R_best;

% Find eigenvalues and eigenvectors:
[eVecsBest,eValsBest] = eig(Lbest,'vector'); % All the evals and evecs
[eValDomBest,Idx] = max(abs(eValsBest)); % Dominant eigenvalue (long-term growth ↙
rate)
eVecDomBest = eVecsBest(:,Idx); % Dominant eigenvector
eVecDomBestNorm = eVecDomBest./(sum(eVecDomBest)); % Norm. dom. evec (long-term ↙
growth distr.)

% Print results:
fprintf('Dominant Eigenvalue (Best Case) = %.4f \n',eValDomBest)
fprintf('Long-Term Growth Rate (Best Case) = %.3f %% \n',(eValDomBest - 1)*100)
fprintf('Long-Term Growth Distribution (Best Case): \n')
```

```
for i = 1:16
    fprintf('Class %.0f = %.4f %% \n',i,eVecDomBestNorm(i)*100)
end

%% Problem 3: 10 Years vs. Long-Term Growth Distribution:

% Find Actual Growth Distribution After 10 Time Steps:
x0 = [0 100 50 50 25 10 zeros(1,10)]'; % Population at Year 0
xBest = (Lbest^10)*x0; % 10 years of best-case growth
xBestNorm = xBest./sum(xBest); % Normalized population distribution

% Print results:
fprintf('Growth Distribution After 10 Years (Best Case): \n')
for i = 1:16
    fprintf('Class %.0f = %.4f %% \n',i,xBestNorm(i)*100)
end

fprintf('Population Count After 10 Years (Best Case): \n')
for i = 1:16
    % Print population of each class, rounded down to the nearest integer:
    fprintf('Class %.0f = %.0f \n',i,floor(xBest(i)))
end

%% Problem 4: Problems 1-3 With Worst-Case Data:

% Build Leslie matrix:
Lworst = diag(S_worst(1:end-1),-1);
Lworst(end,end) = S_worst(end);

Lworst(1,:) = R_worst;

% Find dominant eigenvalue and eigenvector:
[eVecsWorst,eValsWorst] = eig(Lworst,'vector');
[eValDomWorst,Idx] = max(abs(eValsWorst)); % Dominant eigenvalue
eVecDomWorst = eVecsWorst(:,Idx); % Dominant eigenvector
eVecDomWorstNorm = eVecDomWorst./(sum(eVecDomWorst)); % Steady-State Growth↙
Distribution

% Find Actual Growth Distribution After 10 Time Steps:
x0 = [0 100 50 50 25 10 zeros(1,10)]';
xWorst = (Lworst^10)*x0;
xWorstNorm = xWorst./(sum(xWorst));

% Print results:
fprintf('Dominant Eigenvalue (Worst Case) = %.4f \n',eValDomWorst)
fprintf('Long-Term Growth Rate (Worst Case) = %.3f %% \n',(eValDomWorst - 1)*100)
fprintf('Long-Term Growth Distribution (Worst Case): \n')
for i = 1:16
    fprintf('Class %.0f = %.4f %% \n',i,eVecDomWorstNorm(i)*100)
```

```
end

fprintf('Population Count After 10 Years (Worst Case): \n')
for i = 1:16
    % Print population of each class, rounded down to the nearest integer:
    fprintf('Class %.0f = %.0f \n',i,floor(xWorst(i)))
end

fprintf('Growth Distribution After 10 Years (Worst Case): \n')
for i = 1:16
    fprintf('Class %.0f = %.4f %% \n',i,xWorstNorm(i)*100)
end

% Compare Best and Worst Case Results After 10 Years:
subplot(2,1,1)
bBestNorm = bar(xBestNorm.*100);
bBestNorm.FaceColor = 'cyan';
xtips = bBestNorm.XEndPoints;
ytips = bBestNorm.YEndPoints;
labels = string( round(bBestNorm.YData,2) ); % Too cluttered without rounding
labels(bBestNorm.YData < 1) = '< 1';
text(xtips,ytips,↙
labels,'HorizontalAlignment','center','VerticalAlignment','bottom','FontSize',15)
grid on
ylabel('Percentage of Total Population','FontSize',16)
title('Best-Case Growth','FontSize',20)
subplot(2,1,2)
bWorstNorm = bar(xWorstNorm.*100);
bWorstNorm.FaceColor = 'cyan';
xtips = bWorstNorm.XEndPoints;
ytips = round(bWorstNorm.YEndPoints,2);
labels = string( round(bWorstNorm.YData,2) ); % Too cluttered without rounding
labels(bWorstNorm.YData < 1) = '< 1';
text(xtips,ytips,↙
labels,'HorizontalAlignment','center','VerticalAlignment','bottom','FontSize',15)
grid on
xlabel('Age Class','FontSize',16)
ylabel('Percentage of Total Population','FontSize',16)
title('Worst-Case Growth','FontSize',20)

sgtitle('Population Distribution After 10 Years','FontSize',24)

%% Problem 5: Stochastic Growth Rate:

NumSims = 100; % Number of simulations to run
YearsPerTest = 10;
CatChance = 1/25; % Chance of catastrophe occurring

x = zeros(16,YearsPerTest,NumSims); % Initialize x matrix (one row per age class, one ↙
```

```
column per time step, and one frame per simulation)
CatOccur = zeros(NumSims,YearsPerTest); % To keep track of random numbers generated
each simulation
LProd = zeros(16,16,NumSims);

for i = 1:NumSims

    % Keep track of population over 3 variables: time steps, simulations,
    % and age classes:
    for j = 1:YearsPerTest

        CatOccur(i,j) = rand(); % A tragedy can occur any year

        if CatOccur(i,j) <= CatChance
            % If random number (0 to 1) is <= 1/25, then a catastrophe occurs:

            R = 0.7.*R_best; % If a catastrophe year, then R becomes 70% of best-case
        else
            % Else, no tragedy occurs:

            R = R_best;
        end

        L(:,:,j) = diag(S_best(1:end-1),-1); % Build L matrix per year to keep track
of previous catastrophes
        L(end,end,j) = S_best(end);

        L(1,:,j) = R;

        % Time steps are integers from 1 to YearsPerTest, so j signifies time step:

        if j == 1
            LProd(:,:,j) = L(:,:,j);
        else
            LProd(:,:,j) = L(:,:,j)*LProd(:,:,j-1);
        end

        x(:,j,i) = LProd(:,:,j)*x0; % One row per age class, one column per time
step, and one frame per simulation

    end

end

NumCatastrophe = find(CatOccur <= CatChance);
NumCatastrophe = length(NumCatastrophe);
NumNormal = NumSims - NumCatastrophe;

% Plot Results:
```

```
totpop = sum(x,1); % Sum in the rows direction to find total population at each time
step and each simulation
totpop = squeeze(totpop); % Reduce to matrix: one row per time step and one column
per simulation
figure
plot(totpop,'LineWidth',1.5)
xlabel('Time (Years)','FontSize',20)
ylabel('Est. Total Bobcat Population (Counts)','FontSize',20)
title({sprintf('Bobcat Population Growth, %.0f Simulations',NumSims),...
    sprintf('%.0f %% Chance of Catastrophe',CatChance.*100)},'FontSize',24)
grid on
hold on
text(5,500,sprintf('%.0f Catastrophes',NumCatastrophe),'FontSize',20)

TotPopMaxes = max(totpop,[],2);
TotPopMins = min(totpop,[],2);
TotPopSorted = sort(totpop,2,'ascend');

Q1Idx = 0.25*(NumSims + 1);
Q2Idx = 0.50*(NumSims + 1);
Q3Idx = 0.75*(NumSims + 1);

if ~isinteger(Q1Idx)
    % If Q1 location is not an integer, then redefine:
    Q1Idx1 = floor(Q1Idx);
    Q1Idx2 = ceil(Q1Idx);
    d = Q1Idx2 - Q1Idx1;
    TotPopQ1s = TotPopSorted(:,Q1Idx1) + d.*(TotPopSorted(:,Q1Idx2) - TotPopSorted(:,
Q1Idx1));

else
    % If Q1 location is an integer, then don't redefine.
    TotPopQ1s = TotPopSorted(:,Q1Idx);
end
if ~isinteger(Q2Idx)
    % If Q2 location is not an integer, then redefine:
    Q2Idx1 = floor(Q2Idx);
    Q2Idx2 = ceil(Q2Idx);
    d = Q2Idx2 - Q2Idx1;
    TotPopQ2s = TotPopSorted(:,Q2Idx1) + d.*(TotPopSorted(:,Q2Idx2) - TotPopSorted(:,
Q2Idx1));

else
    % If Q2 location is an integer, then don't redefine.
    TotPopQ2s = TotPopSorted(:,Q2Idx);
end
if ~isinteger(Q3Idx)
    % If Q3 location is not an integer, then redefine:
    Q3Idx1 = floor(Q3Idx);
```

```
    Q3Idx2 = ceil(Q3Idx);
    d = Q3Idx2 - Q3Idx1;
    TotPopQ3s = TotPopSorted(:,Q3Idx1) + d.*(TotPopSorted(:,Q3Idx2) - TotPopSorted(:, ↙
Q3Idx1));

else
    % If Q3 location is an integer, then don't redefine.
    TotPopQ3s = TotPopSorted(:,Q3Idx);
end

figure
for i = 1:size(TotPopSorted,1)
    Med = line([i - 0.5 i + 0.5],[TotPopQ2s(i) TotPopQ2s(i)],'LineWidth',↙
2.5,'Color','k');
    Q3 = line([i - 0.3 i + 0.3],[TotPopQ3s(i) TotPopQ3s(i)],'LineWidth',↙
2.5,'Color','b');
    Q1 = line([i - 0.2 i + 0.2],[TotPopQ1s(i) TotPopQ1s(i)],'LineWidth',↙
2.5,'Color','g');
    Mins = line([i - 0.1 i + 0.1],[TotPopMins(i) TotPopMins(i)],'LineWidth',↙
2.5,'Color','r');
    line([i - 0.1 i + 0.1],[TotPopMaxes(i) TotPopMaxes(i)],'LineWidth',↙
2.5,'Color','r')
    line([i i],[TotPopMins(i) TotPopQ1s(i)],'LineWidth',2.5)
    line([i i],[TotPopQ3s(i) TotPopMaxes(i)],'LineWidth',2.5)
    UniquePnts = length(unique(TotPopSorted(i,:))); % Number of unique points per↙
category
    text(i-0.5,TotPopMaxes(i)+100,sprintf('%.0f Unique Points',↙
UniquePnts),'FontSize',12)
end
ylim([0 max(TotPopSorted,[],'all') + 200])
legend([Q1,Q3,Med,Mins],{'25^t^h Percentile','75^t^h Percentile','Median','Minimum /↙
Maximum'},'FontSize',20,'Location','northwest')
grid on
xlabel('Year In Each Simulation','FontSize',20)
ylabel('Est. Population (Counts)','FontSize',20)
title('Spread of Simulated Population Across 10 Years','FontSize',24)
```