```
function [handles] = loadComputeDamageFragilities(handles, filename)
%Load in Fragility Function Information
%Note this only works for fragility and loss function that all have the
%same number of damage states, maybe we could improve?

fragility = readtable(filename);
%[num,txt,raw] = xlsread('SampleFragilityLossFunctionsS.csv') %takes a long
%time but gives an interesting any maybe useful format

nfr = size(fragility,1); %number of fragility curves

% List of the categories of damage fragilities that are inputted, used to
% title variables
categories = table2array(fragility(:,'PG'));
handles.Components = categories;

% Loop through all of the different types of damage fragilities
for i = 1:nfr
    % Find the number DS for each componenet
    handles.(categories{i}).NumDS = (width(fragility(i, :))-2)/4;
    nDam = handles.(categories{i}).NumDS;

    % Pull out all values for a component Fragility
    parse = table2cell(fragility(i, :));
    parameters = zeros(nDam*2,2);
    % Parse through all values for one component, return to better format
    for j = 2:2*nDam+1
        p1 = parse{2*j-1};
        p2 = parse{2*j};
        parameters(j-1, :) = [p1, p2];
    end
    % Save this better formatted damage fragility and loss fragility values
    % to certain component in handles. Use the component title to label the
    % certain variable in handles that we are saving to
    handles.(categories{i}).DSParams = parameters(1:nDam,:);
    handles.(categories{i}).LossParams = parameters(nDam+1:end,:);

    % Save the EDP type to the component
    handles.(categories{i}).EDPtype = parse{2};

    % Assign EDP vector (defined above) to a certain fragility type for
    % later use
    EDP = handles.EDP.(parse{2});
    handles.(categories{i}).EDP = EDP;

    % Initialize DM Fragility. Rows are different DS's, cols are P[DS >
    % ds|EDP]
    handles.(categories{i}).DM_Fragility = zeros(nDam, length(handles.EDP.(parse ↙
{2})));
```

```
    for k = 1:nDam
        median = handles.(categories{i}).DSParams(k, 1);
        sigma = handles.(categories{i}).DSParams(k, 2);
        handles.(categories{i}).DM_Fragility(k,:) = normcdf((log(EDP)-log(median)). ↙
/sigma);
    end

    % Initialize P[Ds = ds | EDP] (subtracting damage fragilities)
    handles.(categories{i}).P_Damage = zeros(nDam+1, length(handles.EDP.(parse{2})));
    handles.(categories{i}).P_Damage(1,:) = 1 - handles.(categories{i}).DM_Fragility ↙
(1,:);

    % Find P[Ds = ds | EDP]
    for z = 1:nDam
        if z < nDam
            handles.(categories{i}).P_Damage(z+1,:) = handles.(categories{i}). ↙
DM_Fragility(z,:) - ...
                                                        handles.(categories{i}). ↙
DM_Fragility(z+1,:);
        else
            handles.(categories{i}).P_Damage(z+1,:) = handles.(categories{i}). ↙
DM_Fragility(z,:);
        end
    end
end

% Examples plots of Damage Fragilities
for k = 1:length(handles.Components)
comp = handles.(handles.Components{k});
legendary = strings(1, comp.NumDS);
figure
hold on
    for i=1:comp.NumDS
        plot(comp.EDP, comp.DM_Fragility(i, :), 'LineWidth', 2)
        legendary(i) = strcat('DS', num2str(i));
    end
    title(strcat(num2str(handles.Components{k}), ' Fragilities'))
    xlabel(strcat('EDP (', comp.EDPtype, ')'))
    ylabel('P[Ds > ds | EDP]')
    legend(legendary)
    set(gca, ...
  'Box'         , 'off'     , ...
  'TickDir'     , 'out'     , ...
  'TickLength'  , [.02 .02] , ...
  'XMinorTick'  , 'on'      , ...
  'YMinorTick'  , 'on');
    grid on
end
```

```
% Display P[DS = ds | EDP]
for k = 1:length(handles.Components)
comp = handles.(handles.Components{k});
legendary = strings(1, comp.NumDS);
figure
hold on
    for i=1:comp.NumDS+1
        plot(comp.EDP, comp.P_Damage(i, :), 'LineWidth', 2)
        if i == 1
            legendary(i) = 'No Damage';
        else
            legendary(i) = strcat('DS', num2str(i-1));
        end
    end
    title(strcat(num2str(handles.Components{k}), ' Probability of being in DS(i)'))
    xlabel(strcat('EDP (', comp.EDPtype, ')'))
    ylabel('P[Ds = ds | EDP]')
    legend(legendary)
    set(gca, ...
  'Box'         , 'off'     , ...
  'TickDir'     , 'out'     , ...
  'TickLength'  , [.02 .02] , ...
  'XMinorTick'  , 'on'      , ...
  'YMinorTick'  , 'on');
    grid on
end
```