  %control to run the next 3 lines of code

clearvars %this just flushes the stack

traps = importTraps(); %loads in the csv file as an object and I can call each column as a variable

types = strcat(traps.InsulationClass,"-",traps.Name); % I am creating a name for the dropdown menu merging 2 columns with a dash in betweem

%control to run the next 26 lines of code

%select Bus

selection = types(7); %drop down menu to select which trap (like for the switches this just finds a column

nameplate = 1; % same as switches

NYtieCAP = false; %110 percent cap applied in NY Tie report %a bool for a for loop later that alters the results if the PJM method is applied, code runs but does nothing to NY method

model = 2; %select which equations to use

%create the range of air temperatures in F for display, but calculations

%must be done in C

AT = f2c(-40:5:125); %from -40 to 125F in 5F steps, create an array and convert it from deg f to def c

TempRangeLabels = -40:5:125;

He = 0; %elevation in meters

%Conductor parameters

index = find(strcmp(types,selection)); %pull in the MOT and Rise from the sheet and show on the screen like switches

MOT = traps.MOT(index);

Rise = traps.Rise(index);

%these are flags for later corrections

if He < 1000 %should probably be a switch but this works

AltitudeFactor = 1;

MaxAT = 40;

elseif He < 1500

AltitudeFactor = 0.99;

MaxAT = 37;

else

AltitudeFactor = 0.96;

MaxAT = 30;

end

**PJM Method**

not in IEEE guide

if model == 1

n=2; %this is the n constant we had before it does not need to be displayed

%modified it with a catch for AT above MOT

delta = MOT-AT; %the MOT constant minus the air temp array,

I = zeros(numel(delta), 1); %creates a placeholder array for the result "Rating in Amps"

I(delta > 0) = nameplate .\* ((MOT - AT(delta > 0)) ./ Rise).^(1./n); %for every AT provided, if delta is positive, then complete the equation

I(delta <= 0) = 0; %for all columns where delta is zero or negative set the rating to zero

else

**NY-Tie Method**

not in IEEE guide

load("TIE\_curve.mat"); %had to create a custom curve because they dont have an equation

tempsF = curve(1,:); %degrees F as a column extracted from their plot

factors = curve(2,:) ./ 100; %convert from percentage to a multiplier - this gets multiplied by nameplate, column extracted from their plot

for i =1:numel(AT)

I(i) = interp1(tempsF, factors, c2f(AT(i)),'spline'); %If the air temperature is a value between two points in their curve, interpolate the value, linear fit is fine too

end

if max(AT > 40) %I extended the curve they provided, we want to warn against using my data in that range

disp("Data extrpolated beyond known curve - may contain errors")

end

end

**Elevation Factor**

is in standard but not in PJM or NY guides

I = I .\* AltitudeFactor; %from the earlier elseif block

**Rating**

in amps

if NYtieCAP

I(I > 1.1 \* nameplate) = 1.1 .\* nameplate; %if cap checkbox selected, for all values where rating(I) is more than 110% of the nameplate, set it to 110% of the nameplate

end

if max(AT)> MaxAT %if peak air temperature in our range goes above the allowed correction curve throw a warning

disp("Warning, may be above valid range for local altitude")

end

if nameplate < 100 %

I = round(I,2,'significant') % round to 2 decimal places if nameplate is under 100

else

I = round(I) %round to 0 decimal places otherwise

end

%--- plot data as for switches

Import Trap Parameters

%this code wont matter to you, its just for importing the excel sheet

function [trapTable] = importTraps()

opts = delimitedTextImportOptions("NumVariables", 6, "Encoding", "UTF-8");

% Specify range and delimiter

opts.DataLines = [2, Inf];

opts.Delimiter = ",";

% Specify column names and types

opts.VariableNames = ["Name", "InsulationClass", "Rise", "MOT", "EmergencyGreaterThan24H", "Emergency24HOrLess"];

opts.VariableTypes = ["string", "string", "double", "double", "double", "double"];

% Specify file level properties

opts.ExtraColumnsRule = "ignore";

opts.EmptyLineRule = "read";

% Specify variable properties

opts = setvaropts(opts, ["Name", "InsulationClass"], "WhitespaceRule", "preserve");

opts = setvaropts(opts, ["Name", "InsulationClass"], "EmptyFieldRule", "auto");

% Import the data

trapTable = readtable("C:\Users\pjbe002\Documents\MATLAB\DTCR\trapTable.csv", opts);

end