%% MATLAB sprinkle system uniformity script

millilitertocubicinches = 0.061024; % volume measurement

catchcanarea = 24.30132; % sq inches

% IMPORT DATA SET FROM EXCEL - labeled "data" in the Workspace.

% ROTATE DATA AND CONVERT VOLUME TO DEPTH

% data % shows original volume matrix.

% rotates original data set so that sprinkler lateral is vertical

% NOTE: create if/else statement to make future replications more automated

data1 = rot90(data);

% data1 = data % use this if original data set is taken along vertical

% sprinkler lateral.

% convert volumes to depths

depth = data1\*millilitertocubicinches/catchcanarea; % converting volume to

% depth, in

% RADIAL CANS

% extract values corresponding to radial cans

radialdistance = 0:10:40; % distance from sprinkler, ft

radialcans = fliplr(depth(end,1:5)); % Extract 1-5 values from last row, and

% flips them left to right so that 0 distance corresponds with can adjacent

% to sprinkler

p = polyfit(radialdistance,radialcans,2); % creates three value array with

% coefficients of a x^2 fit polynomial equation

% create meshgrid, which is a representation of area.

x = -40:10:40; % radius perpendicular to the lateral with sprinkler in middle.

y = 0:10:40; % DISTANCE FROM SPRINKLER ALONG LATERAL TO FIELD EDGE

[X,Y] = meshgrid(x,y);

% create distance vector (i.e the distance from each cell to the sprinkler)

d = sqrt(X.^2+Y.^2);

% create swept radial matrix on SOUTH of field

southedge = nan\*ones(size(d)); % create matrix size d with NaN

southedge(d>=50)=0; % when distance is >= 50, zero depth

southedge(d<50)=p(1)\*d(d<50).^2+p(2)\*d(d<50)+p(3); % % for radius within reach,

% use polynomial equation

% create swept radial matrix on NORTH of field

northedge = rot90(rot90(southedge));

% delete first row (outside of bounds)

northedge([1],:) = [];

% DEFINING FIELD DIMENSIONS

fieldlength = 150; % field length along lateral, ft

fieldwidth = 130; % field width along lateral, ft

field = zeros(fieldlength/10+1,fieldwidth/10+1); % create array of field

% area with dimensions in 10 ft increments

% specifying number of rows and columns in the field

[rowsField, columnsField] = size(field);

[rowsdepth, columnsdepth] = size(depth);

% PASTING A SMALL MATRIX WITHIN A LARGER ONE

% !!!DO THIS AS MANY TIMES AS THERE ARE CatchCan Data sets along lateral

% PASTING FIRST DEPTH MATRIX INTO FIELD

% Specify upper left row, column of where we'd like to paste the small matrix.

row1 = 9; %

column1 = 1;

% Determines lower right location.

row2 = row1 + rowsdepth - 1;

column2 = column1 + columnsdepth - 1;

% See if it will fit.

if row2 <= rowsField

% It will fit, so paste it.

field(row1:row2, column1:column2) = depth; % field array now includes

% first matrix

else

% It won't fit

warningMessage = sprintf('That will not fit',...

row2, column2);

uiwait(warndlg(warningMessage));

end

% PASTING SECOND DEPTH MATRICES INTO FIELD

% Specify upper left row, column of where we'd like to paste the small matrix.

row1 = 5; %

column1 = 1;

% Determines lower right location.

row2 = row1 + rowsdepth - 1;

column2 = column1 + columnsdepth - 1;

% See if it will fit.

if row2 <= rowsField

% It will fit, so paste it.

field(row1:row2, column1:column2) = depth;

else

% It won't fit

warningMessage = sprintf('That will not fit',...

row2, column2);

uiwait(warndlg(warningMessage));

end

% REPEAT PASTING OF DEPTH MATRICES UNTIL ENTIRE LENGHT OF LATERAL HAS

% REPRESENTATIVE VALUES

% DELETE VALUES IN LAST ROW OF LATERAL DEPTH MATRIX

field([12],:) = [0];

% ADD SOUTHEDGE WITH LATERAL VALUES

[rowssouthedge, columnssouthedge] = size(southedge);

% Specify upper left row, column of where we'd like to paste the small matrix.

row1 = 12; %

column1 = 1;

% Determines lower right location.

row2 = row1 + rowssouthedge - 1;

column2 = column1 + columnssouthedge - 1;

% See if it will fit.

if row2 <= rowsField

% It will fit, so paste it.

field(row1:row2, column1:column2) = southedge;

else

% It won't fit

warningMessage = sprintf('That will not fit',...

row2, column2);

uiwait(warndlg(warningMessage));

end

% ADD NORTHEDGE WITH LATERAL VALUES

[rowsnorthedge, columnsnorthedge] = size(northedge);

% Specify upper left row, column of where we'd like to paste the small matrix.

row1 = 1; %

column1 = 1;

% Determines lower right location.

row2 = row1 + rowsnorthedge - 1;

column2 = column1 + columnsnorthedge - 1;

% See if it will fit.

if row2 <= rowsField

% It will fit, so paste it.

field(row1:row2, column1:column2) = northedge;

else

% It won't fit

warningMessage = sprintf('That will not fit',...

row2, column2);

uiwait(warndlg(warningMessage));

end

% CIRCSHIFT(FIELD) - move lateral across field.

% The rectangular field area (not actual field shape) starts at column 3

FirstPipeDistance = 10; % distance from left edge of selected area to lateral

NoOfCansInRadial = 5; % number of cans in radial, including can at sprinkler

initialshift = -(NoOfCansInRadial-(FirstPipeDistance/10+1));

field1 = circshift(field, [0, initialshift]);

distancebetweenmoves = 60; % distance to apply circshift

field2 = circshift(field1, [0, distancebetweenmoves/10]);

field3 = circshift(field2, [0, distancebetweenmoves/10]);

% delete out-of-bound columns - I could automate this process, but it seems

% unnecessary as it only needs to be done to first and last move at most.

field1(:,[12,13,14]) = [0]; % deletes initial columns that fall outside area

field3(:,[1,2,3,4]) = [0]; % deleted final columns that fall outside of area

fieldfinal = field1 + field2 + field3;

%surf(fieldfinal);

figure

% CREATE 3D IMAGE OF FIELD DEPTHS

surf(flipud(fieldfinal)) % flips fieldfinal so that surface image is as

% appears from observer in field.

axis equal % sets aspect ratio of axis to equal

% CALCULATE CU

m = mean(fieldfinal(:));

abszm = abs(fieldfinal-m);

CU = 100\*(1-sum(abszm(:))/sum(fieldfinal(:)))

% CALCULATE DU

a = 0.25\*numel(fieldfinal); % number of elements in low quarter

b = sort(fieldfinal(:)); % sorts matrix elements in ascending order

lowqvalues = b(1:a); % Extract 1-5 values from last row, and flips them left

% to right so that 0 distance corresponds with can adjacent to sprinkler

DU = 100\*mean(lowqvalues)/mean(fieldfinal(:))