Table of Contents

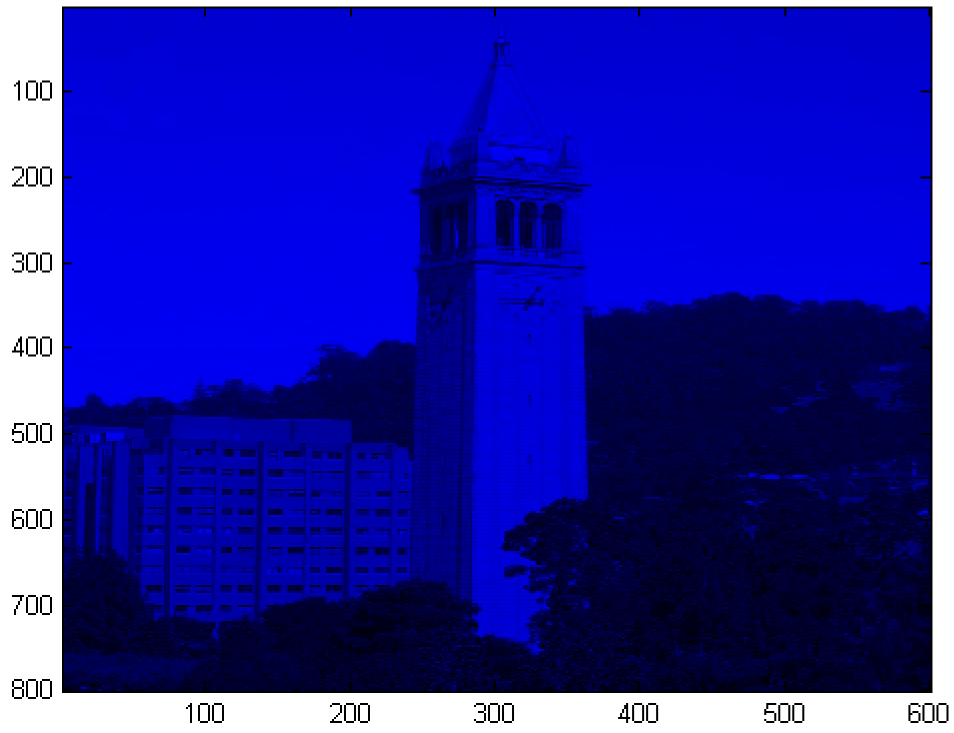
E7 Lab 7 Solutions

Spring 2016

```
format compact
format short
clear all
clc
close all
```

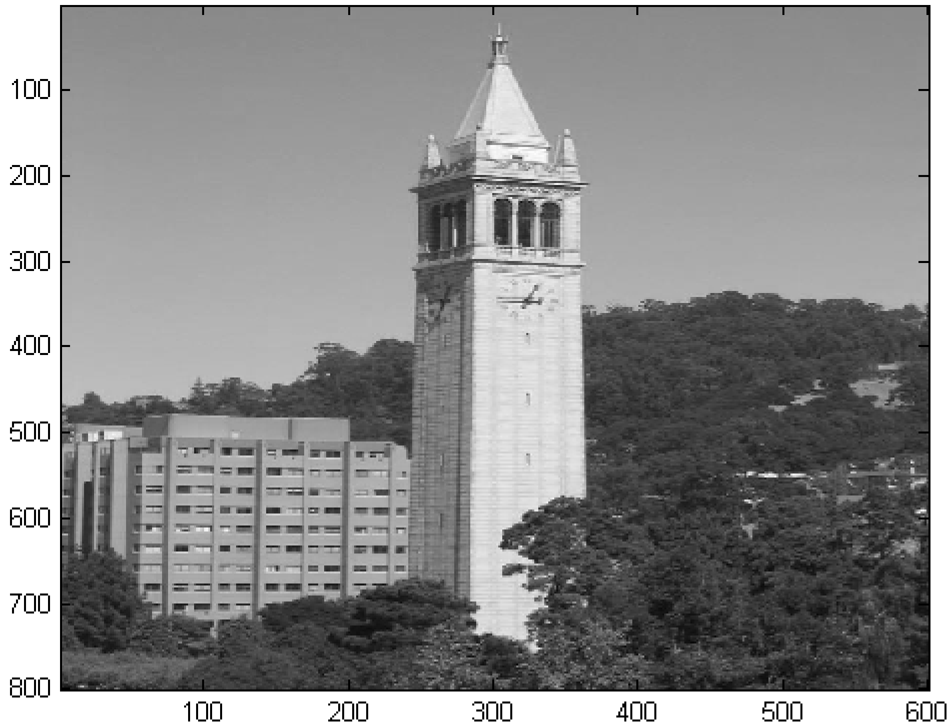
testing problem 1

```
type myRGBDecomposition

img = double(imread('sather.jpg'))/255;
[img_red, img_green, img_blue] = myRGBDecomposition(img);
image(img_red);
image(img_green);
image(img_blue);



function [img_red, img_green, img_blue] = myRGBDecomposition( img )
%decomposes an image into its red, green, and blue bands.
%img: matlab image object
%img_red: extracted red band
%img_green: extracted green band
%img_blue: extracted blue band
rows = size(img,1);
cols = size(img,2);

img_red = zeros(rows,cols,3);
img_green = zeros(rows,cols,3);
img_blue = zeros(rows,cols,3);

img_red(:,:,1) = img(:,:,1);
img_green(:,:,2) = img(:,:,2);
img_blue(:,:,3) = img(:,:,3);
end
```



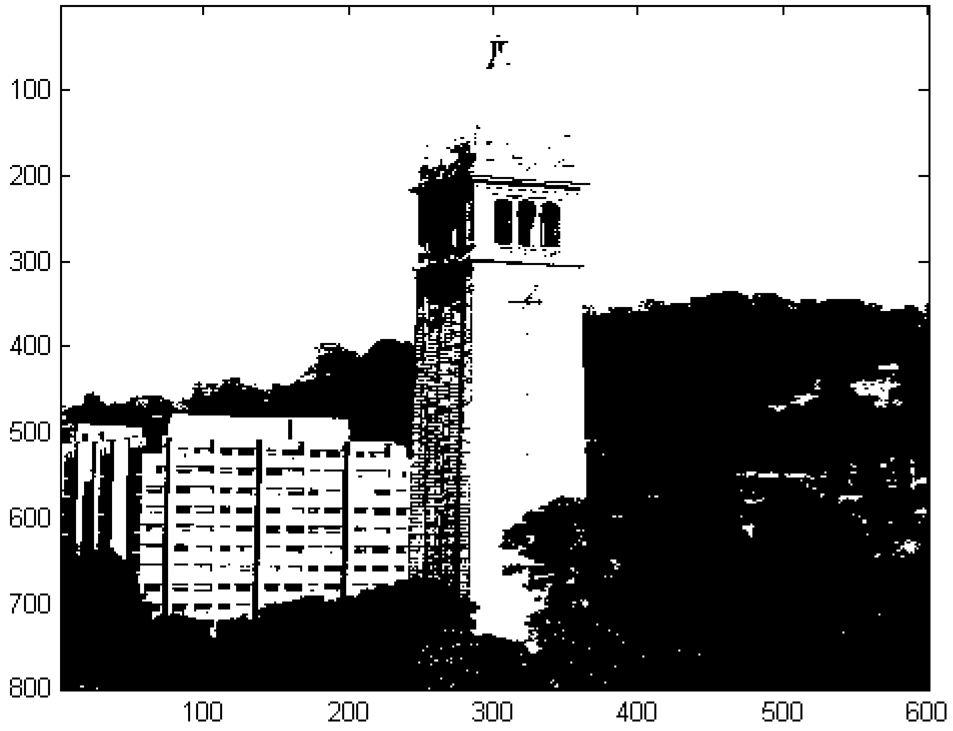
testing problem 1.2

```
type myGrayConverter

img = double(imread('sather.jpg'))/255;
[img_gray] = myGrayConverter(img);
image(img_gray);



function [img_gray] = myGrayConverter(img)
%decomposes an image into its grayscale equivalent.
%img: matlab image object
%img_gray: converted grayscale image

pixelMeans = (img(:,:,1)+img(:,:,2)+img(:,:,3))/3;
img_gray = repmat(pixelMeans,[1,1,3]);


end
```



problem 1.3

```
type myBinaryConverter

img = double(imread('sather.jpg'))/255;
[img_gray] = myGrayConverter(img);
img_binary = myBinaryConverter(img_gray,.5);
image(img_binary);



function [img_binary] = myBinaryConverter(img_gray, threshold)
%converts a grayscale image into a black and white image.
%img_gray: grayscale matlab image object
%threshold: grayscale pixels above threshold are converted to 1
%others converted to zero
%img_binary: converted black and white image

img_binary = zeros(size(img_gray,1),size(img_gray,2),3);
img_binary(img_gray>threshold)=1;


end
```



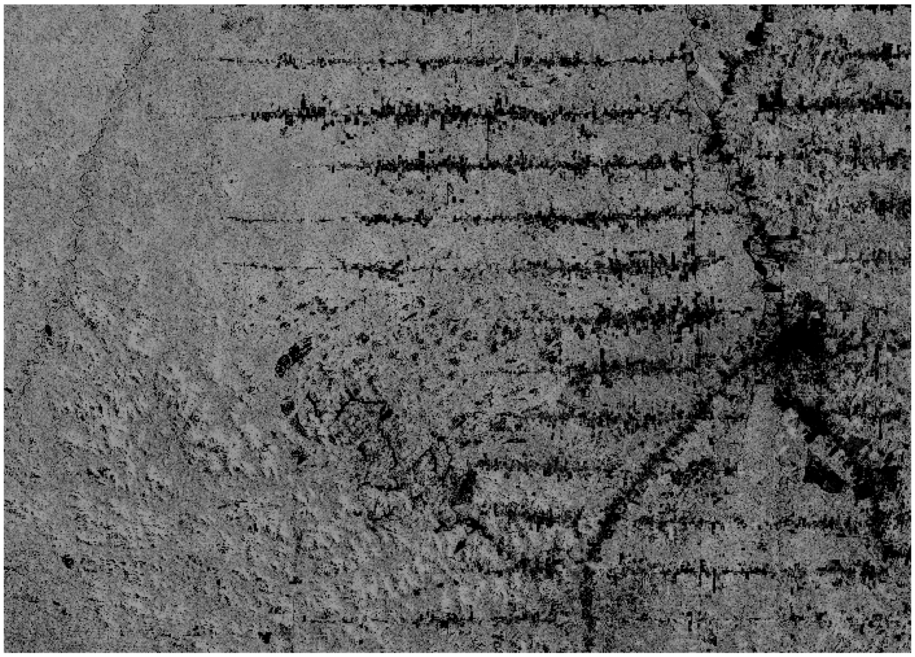
problem 1.4

```
type myVintageFilter

img = double(imread('sather.jpg'))/255;
[img_vintage] = myVintageFilter(img);
imshow(img_vintage);



function [img_vintage] = myVintageFilter(img)
%apply vintage filter to an image
%img: matlab image object
%img_vintage: filtered vintage image
img_vintage = zeros(size(img));
redOld = img(:,:,1); greenOld = img(:,:,2); blueOld = img(:,:,3);

img_vintage(:,:,1) = .393*redOld + .769*greenOld + .189*blueOld;
img_vintage(:,:,2) = .349*redOld + .686*greenOld + .168*blueOld;
img_vintage(:,:,3) = .272*redOld + .534*greenOld + .131*blueOld;

end


Warning: Image is too big to fit on screen; displaying at
67%
```

problem 2.1

```
type myNDVI

aug_rgb = double(imread('brazil_1985_Aug_rgb.png'))/255;
aug_nir = double(imread('brazil_1985_Aug_nir.png'))/255;
ndvi = myNDVI(aug_rgb,aug_nir);
imshow(ndvi);



function [NDVI] = myNDVI(img_RGB, img_NIR)
%calculate whether a pixel of a satellite
%image corresponds to a vegetated
%ared or not using the pixel's Normalized
%Difference Vegetation Index
```

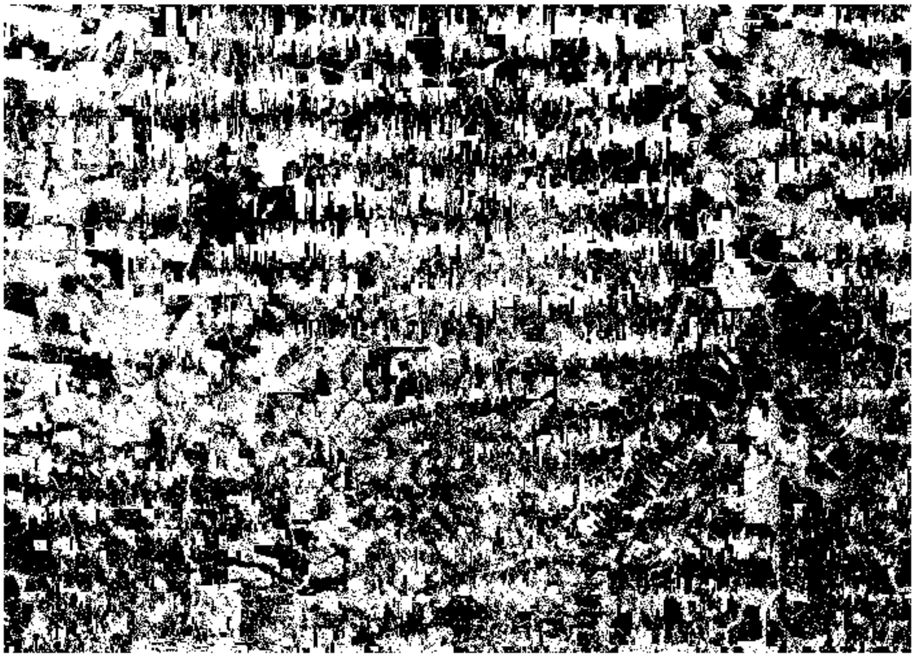
```
%(NDVI) = (NIR-RED)/(NIR+RED)
%NDVI: NxM array containing NDVI calculated
%for each pixel
%img_RGB: NxMx3 array for visible image
%img_NIR: NxMx3 array for near-infrared image
A = img_RGB(:,:,1);
B = img_NIR(:,:,1);
epsilonA = min(min(A(A>0)));
epsilonB = min(min(B(B>0)));
img_RGB(img_RGB==0) = epsilonA;
img_NIR(img_NIR==0) = epsilonB;

RED = img_RGB(:,:,1);
NIR = img_NIR(:,:,1);

NDVI = (NIR - RED)./(NIR + RED);


end
```



problem 2.2

```
type vegArea

aug_rgb = double(imread('brazil_1985_Aug_rgb.png'))/255;
aug_nir = double(imread('brazil_1985_Aug_nir.png'))/255;
[veg_area, img_veg] = vegArea(aug_rgb, aug_nir, .15);
veg_area
imshow(img_veg);
```

```
aug_rgb = double(imread('brazil_2015_Aug_rgb.png'))/255;
aug_nir = double(imread('brazil_2015_Aug_nir.png'))/255;
[veg_area, img_veg] = vegArea(aug_rgb, aug_nir, .15,88.9);
veg_area
imshow(img_veg);



function [veg_area, img_veg] = vegArea(img_RGB, img_NIR,...
    threshold, varargin)
%computes the total area of the image classified as vegetates,
%as determined by an inputted NDVI threshold
%veg_area: scalar double, total vegetated area. Without width it
%represents percentage of total surface area. If Width defined,
%expressed as area in km^2
%img_veg: binary NxM array for vegetation classigication
%(vegetated = 1, otherwise = 0)
%img_RGB: NxMx3 array corresponding to visible image
%img_NIR: NxMx3 array corresponding to near-infrared image
%threshold: inputted NDVI threshold
%varargin: additional optional input, width (real-world width
% of the region [km])
ndvi = myNDVI(img_RGB,img_NIR);
img_veg = zeros(size(ndvi));
img_veg(ndvi>=threshold) = 1;

veg_area = numel(img_veg(img_veg==1))/numel(img_veg);

if ~isempty(varargin)
    width = varargin{1};
    pixelWidth = width/size(ndvi,2);
    image_area = (pixelWidth*size(ndvi,2))*(pixelWidth*size(ndvi,1));
    veg_area = veg_area*image_area;
    return;
end

veg_area=veg_area*100;

end


veg_area =
   84.0457
veg_area =
  2.5477e+003
```



problem 2.3

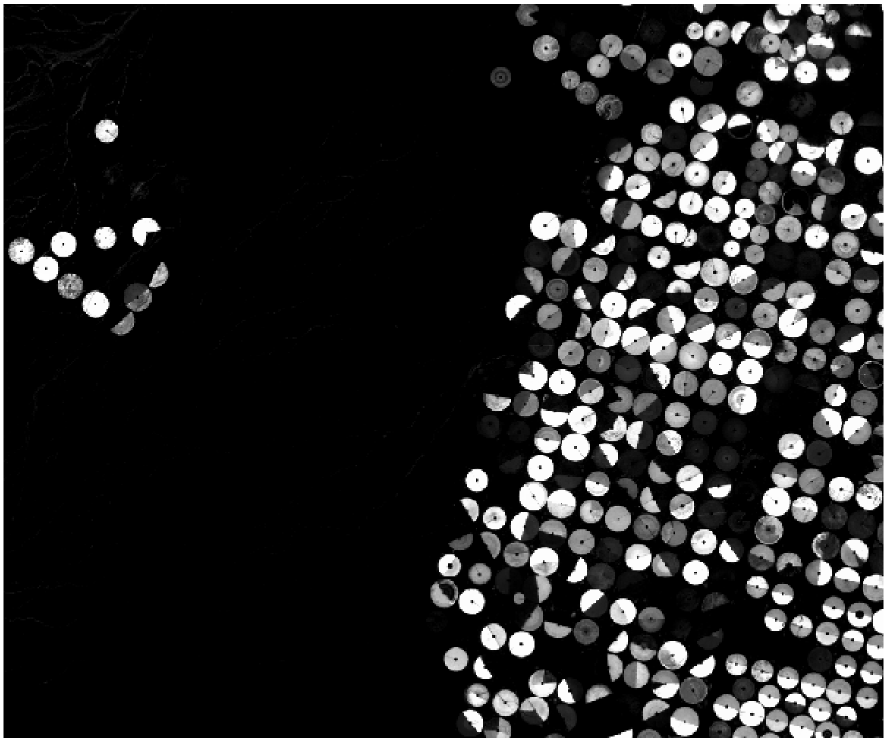
```
type vegChange

img_RGB1 = double(imread('brazil_1985_Aug_rgb.png'))/255;
img_NIR1 = double(imread('brazil_1985_Aug_nir.png'))/255;
img_RGB2 = double(imread('brazil_2015_Aug_rgb.png'))/255;
img_NIR2 = double(imread('brazil_2015_Aug_nir.png'))/255;
veg_diff = vegChange(img_RGB1, img_NIR1 , img_RGB2, img_NIR2 , .15, 88.9);
veg_diff
```

```
function [veg_diff] = vegChange(img_RGB1, img_NIR1, img_RGB2, img_NIR2,threshold,
%takes two sets of RGB and NIR images representing the same area at
%different times and returns the total vegetation change in km^2, given a
%NDVI threshold and real-world width of the region.
%assumes first image is the oldest
[veg_area1, img_veg1] = vegArea(img_RGB1, img_NIR1, threshold, width);
[veg_area2, img_veg2] = vegArea(img_RGB2, img_NIR2, threshold, width);

veg_diff = veg_area2 - veg_area1;

end
```

```
veg_diff =
 -2.1968e+003
```

problem 2.1

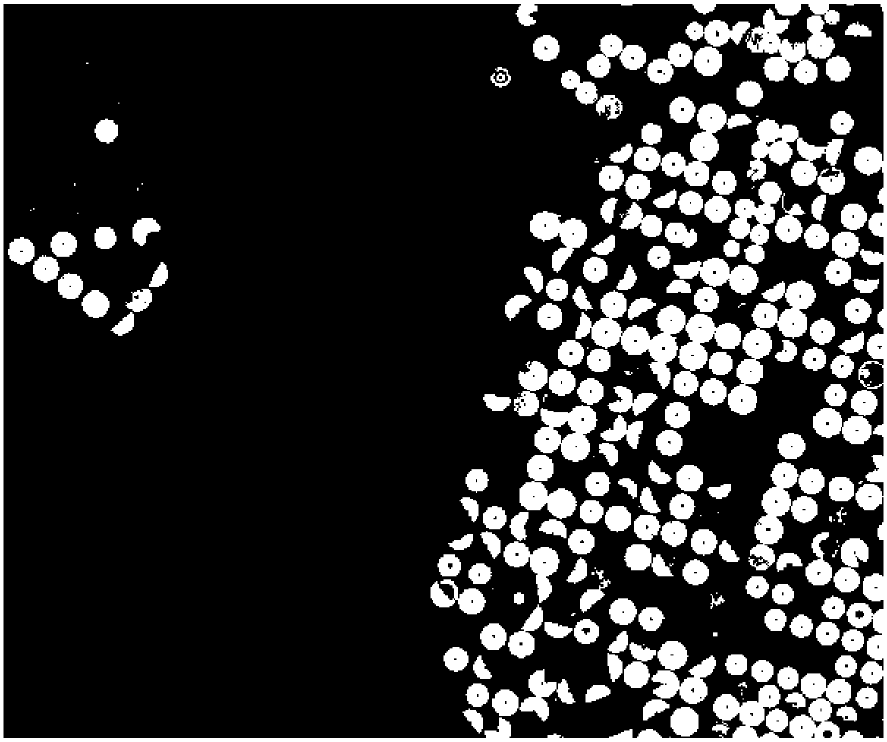
```
dec_rgb = double(imread('saudi_arabia_2015_Dec_rgb.png'))/255;
dec_nir = double(imread('saudi_arabia_2015_Dec_nir.png'))/255;
ndvi = myNDVI(dec_rgb,dec_nir);
imshow(ndvi);
```



problem 2.2

```
dec_rgb = double(imread('saudi_arabia_2015_Dec_rgb.png'))/255;
dec_nir = double(imread('saudi_arabia_2015_Dec_nir.png'))/255;
[veg_area, img_veg] = vegArea(dec_rgb, dec_nir, .2);

imshow(img_veg);
veg_area
```

```
veg_area =
   17.9997
```



problem 2.3

```
img_RGB1 = double(imread('saudi_arabia_1984_Dec_rgb.png'))/255;
img_NIR1 = double(imread('saudi_arabia_1984_Dec_nir.png'))/255;
img_RGB2 = double(imread('saudi_arabia_2015_Dec_rgb.png'))/255;
img_NIR2 = double(imread('saudi_arabia_2015_Dec_nir.png'))/255;

veg_diff = vegChange(img_RGB1, img_NIR1 , img_RGB2, img_NIR2 , .2, 88.9)
```

```
veg_diff =
  1.1855e+003
```

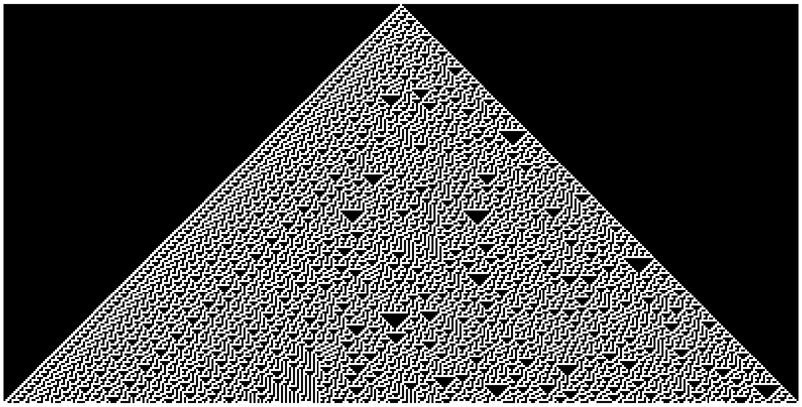
problem 3

```
type myCellAuto

im_big = myCellAuto(30,200);
imshow(im_big)
im_small = myCellAuto(30,5);
im_small
```

```
function [pattern] = myCellAuto(rule,step)
%plots a black and white image using cell automation given by input rule
%for the number of timesteps step
```

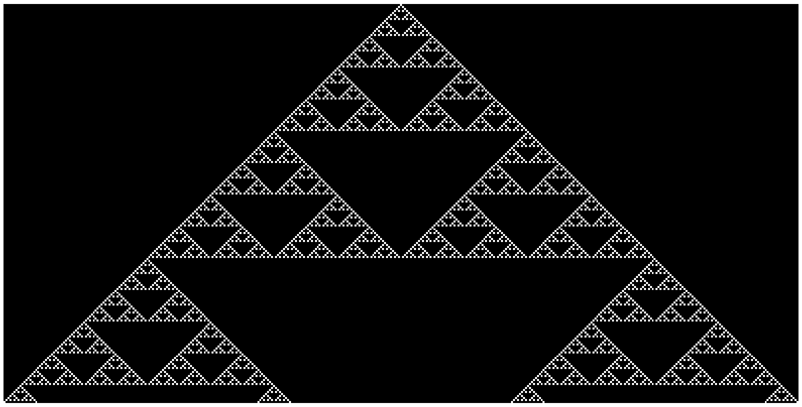
```
%rule: binary representation of number that determines the next state of a
%value depending on its neighbor state
%step: the number of timesteps to iterate through. The resulting image's
%final width is dependent on the number of timesteps. For N timesteps there
%should be N+1 rows and 2*N+1 columns in the image.
N = step;
pattern = zeros(N+1,2*N+1);
pattern(1,ceil((2*N+1)/2))=1;
rule = dec2bin(rule);
while length(rule)<8
    rule = ['0' rule];
end

flippedRule = fliplr(rule);

for row = 2:size(pattern,1)
    pattern(row,1) = str2num(flippedRule(toDec([0 pattern(row-1,1:2)])));
    pattern(row,end) = str2num(flippedRule(toDec(...
        [pattern(row-1,(end-1):end) 0])));
    for col = 2:(size(pattern,2)-1)
        pattern(row,col) = str2num(flippedRule(toDec(...
            pattern(row-1,(col-1):(col+1)))));
    end
end

%imshow(pattern)
% commented out to control number of plots on published file!
end


function [dec] = toDec(binary)
dec = 2^2*binary(1)+2*binary(2)+binary(3)+1;
end
```

```
im_small =
  Columns 1 through 10
     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     1     1     1     0     0     0
     0     0     0     1     1     0     0     1     0     0
     0     0     1     1     0     1     1     1     1     0
     0     1     1     0     0     1     0     0     0     1
     1     1     0     1     1     1     1     0     1     1
  Column 11
     0
     0
     0
     0
     0
     1
```



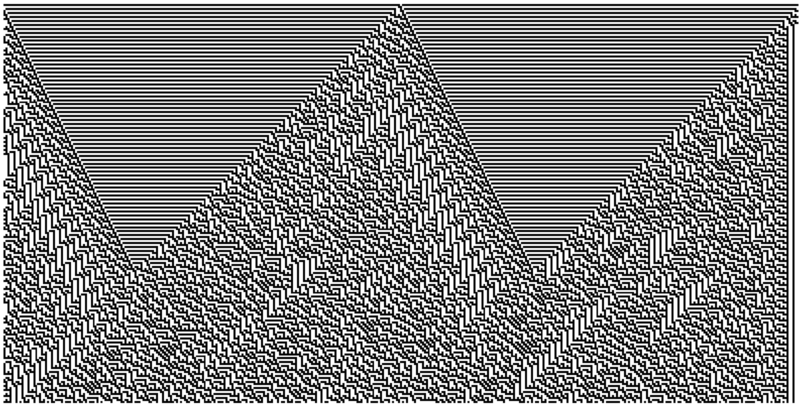
```
im_big = myCellAuto(146,200);
imshow(im_big)
im_small = myCellAuto(146,5);
im_small
```

```
im_small =
  Columns 1 through 10
     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     1     0     1     0     0     0
     0     0     0     1     0     0     0     1     0     0
     0     0     1     0     1     0     1     0     1     0
     0     1     0     0     0     0     0     0     0     1
     1     0     1     0     0     0     0     0     1     0
  Column 11
     0
     0
     0
     0
     0
     1
```



```
im_big = myCellAuto(89, 200);
imshow(im_big)
im_small = myCellAuto(89, 5);
im_small
```

```
im_small =
  Columns 1 through 10
     0     0     0     0     0     1     0     0     0     0
     1     1     1     1     0     0     1     1     1     1
     1     0     0     1     1     0     1     0     0     0
     0     1     0     1     1     0     0     1     1     0
     0     0     0     1     1     1     0     1     1     1
     1     1     0     1     0     1     0     1     0     0
  Column 11
     0
     1
     1
     0
     1
     1
```



Published with MATLAB® 7.10