22S-CSB150-EEB159 Lab9

LISA WANG

TOTAL POINTS

100 / 100

QUESTION 1

1 Ex.1 **40 / 40**

✓ **- 0 pts** *Correct*

- 10 pts Correctly submit greyscale image

- 10 pts Plot with 3 lines

- 10 pts state that both are close to empirical solution

- 10 pts Correctly state that local slope analysis is better

- 2 pts invalid explanation

QUESTION 2

2 Ex.2 **30 / 30**

✓ **- 0 pts** *Correct*

- 5 pts Greyscale image

- 3 pts Plot with 3 lines

- 10 pts Both quite close to empirical solution

- 3 pts Local slope analysis is better + valid explanation

QUESTION 3

3 Ex.3 **30 / 30**

✓ **- 0 pts** *Correct*

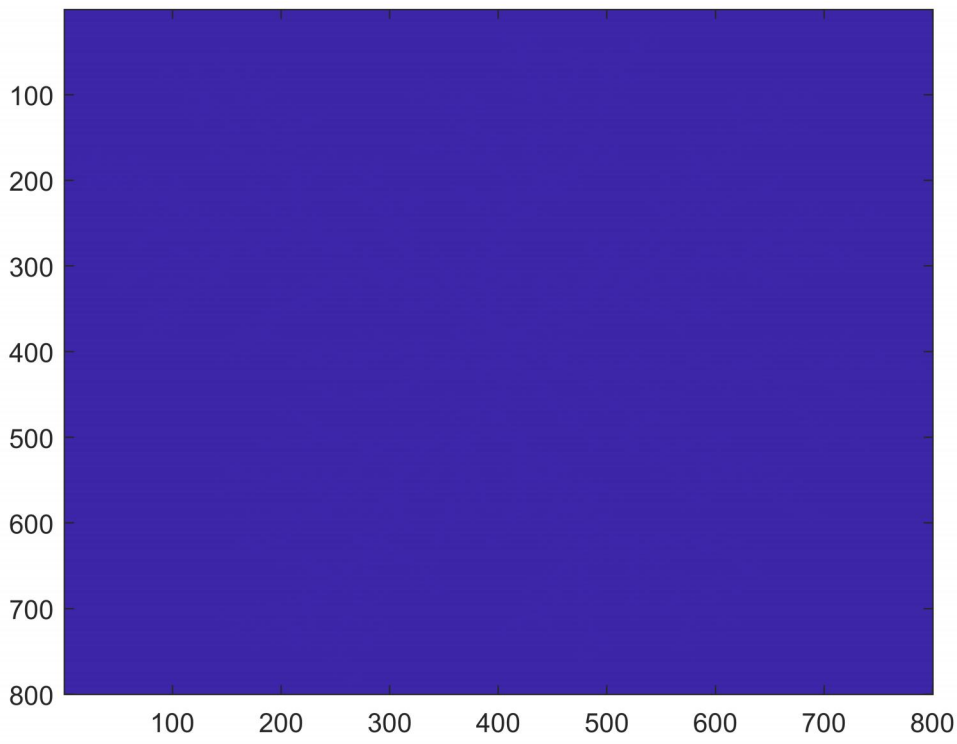
- 5 pts Grayscale image

- 3 pts Plot with 3 lines

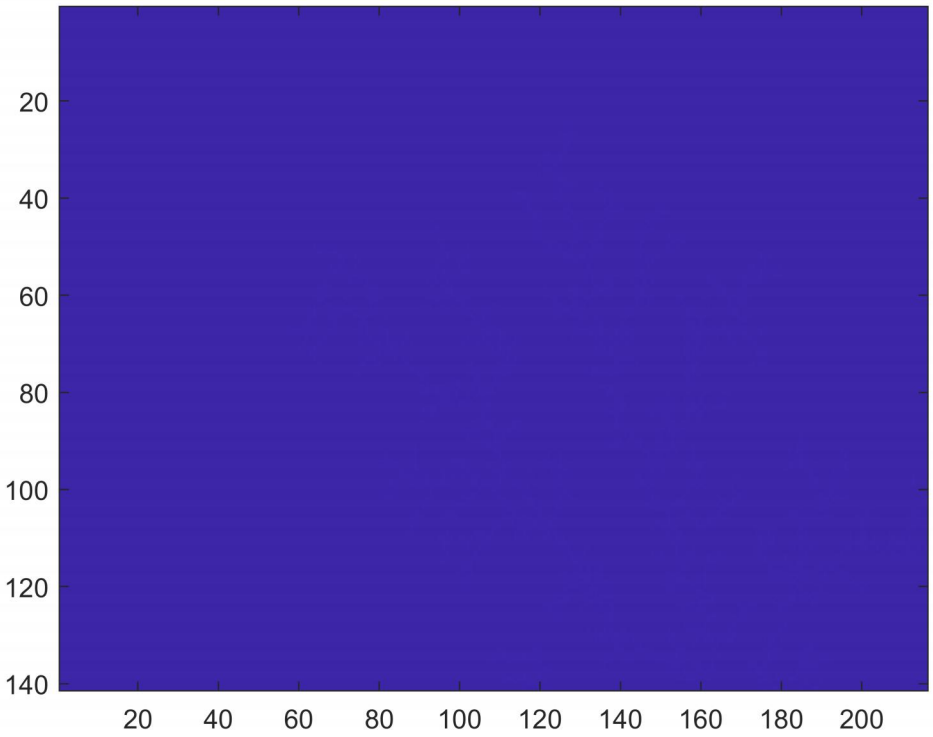
- 10 pts both close to empirical solution

- 5 pts Local slope better or sufficient explanation

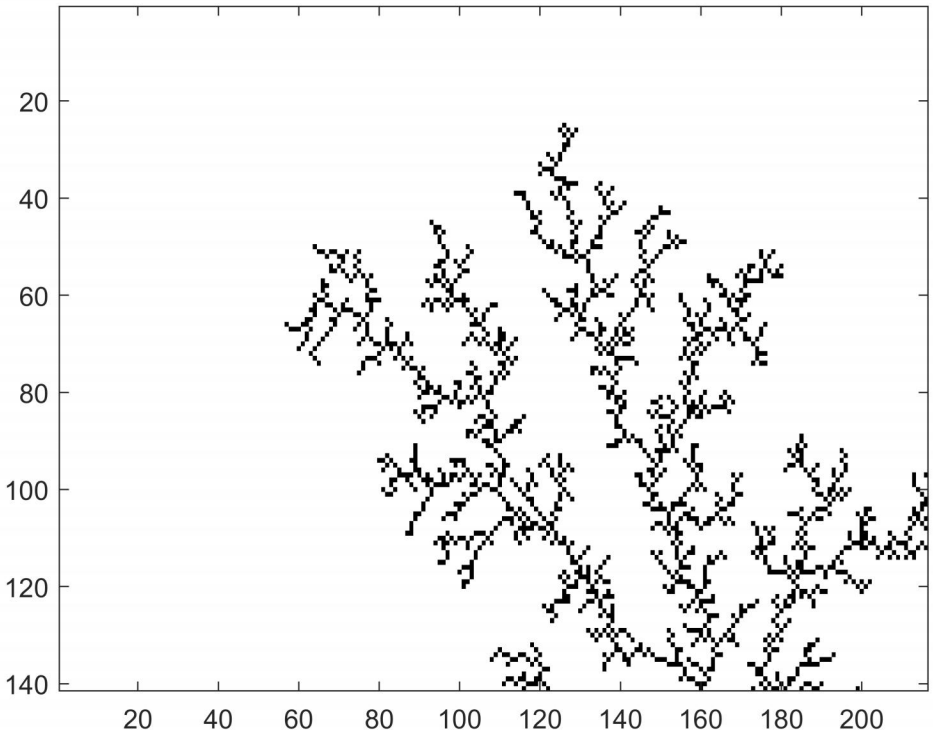
```
% lab 9

%(loads image file)
im1 = imread('dla.gif');

%(Look at image)
image(im1);
```



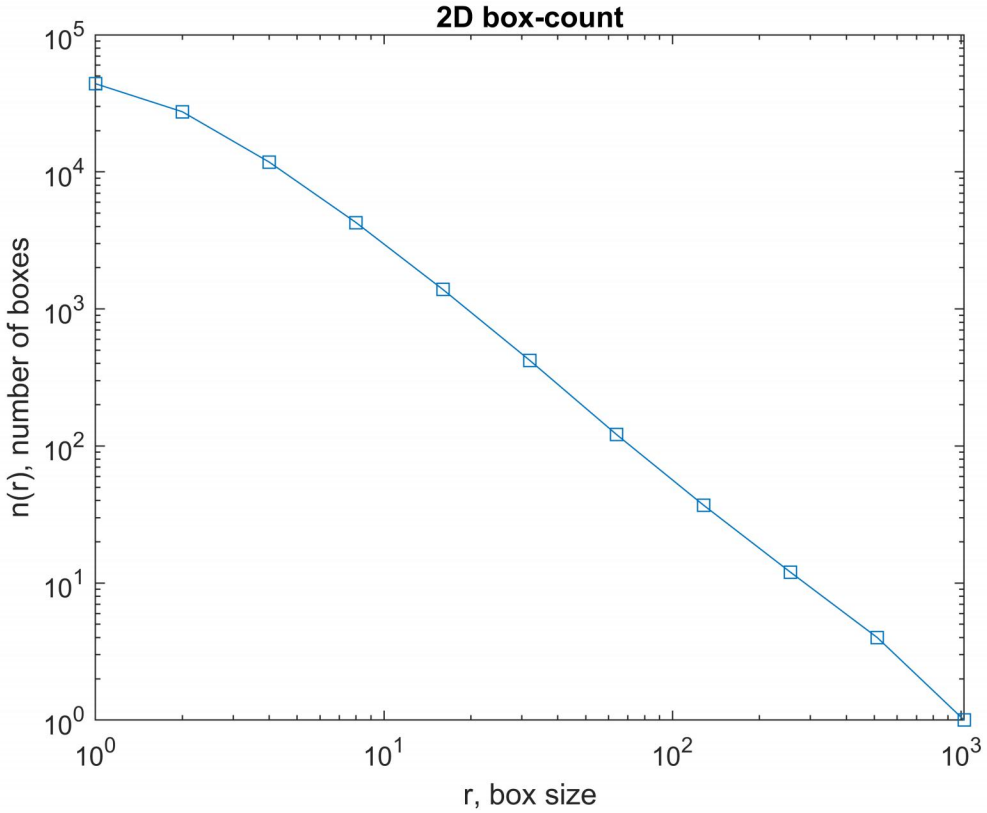

```
%(pull out just part you want)
i = im1(20:160, 25:240, 1);

%(Check what you pulled out)
image(i)
```




```
%(Sets full color scale for image)
imagesc(~i);
%(Converts to gray scale)
colormap gray;
```



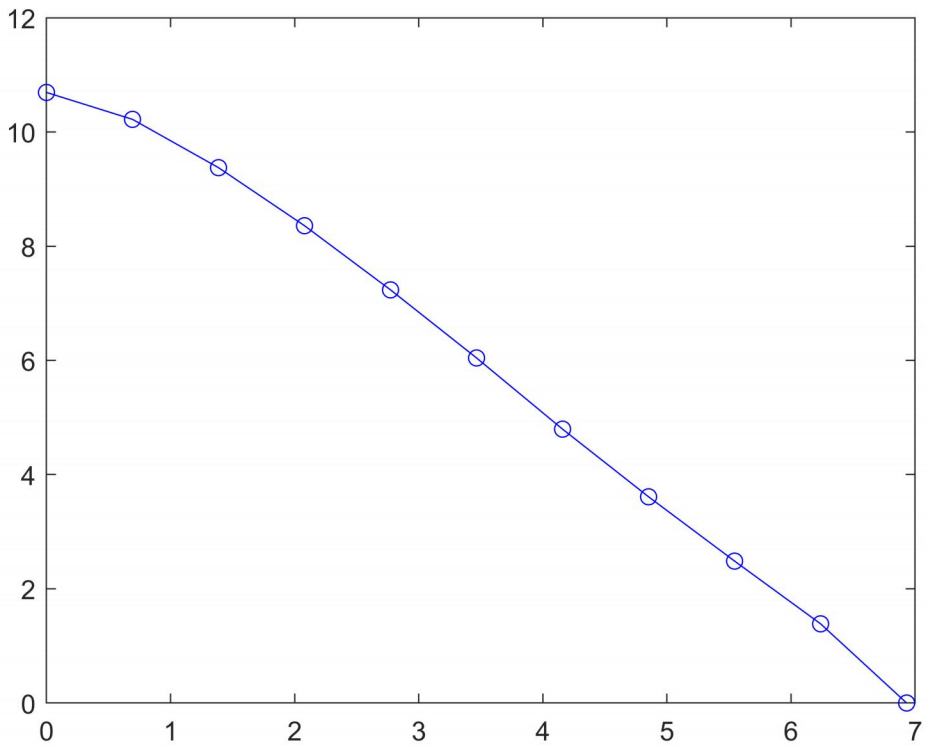

```
%(run the command and generate the data)
boxcount(im1)
```



```
%(stores data in matrix for n=number of boxes, r=size/radius of boxes)
[n,r]=boxcount(im1)
```

```
n = 1×11
       44000       27466       11786        4265        1386         421 ···
r = 1×11
           1           2           4           8          16          32 ···
```

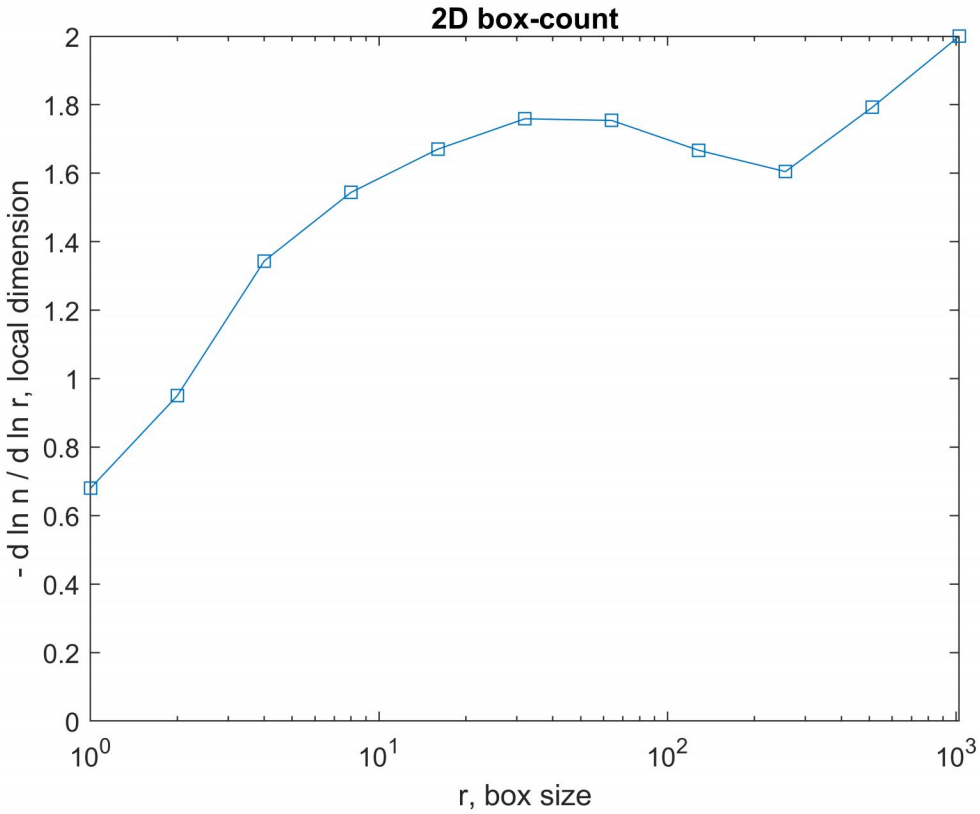
```
%(look at data, x axis is 1dt argument, y axis is 2nd argument)
plot(log(r),log(n),'bo-')
```



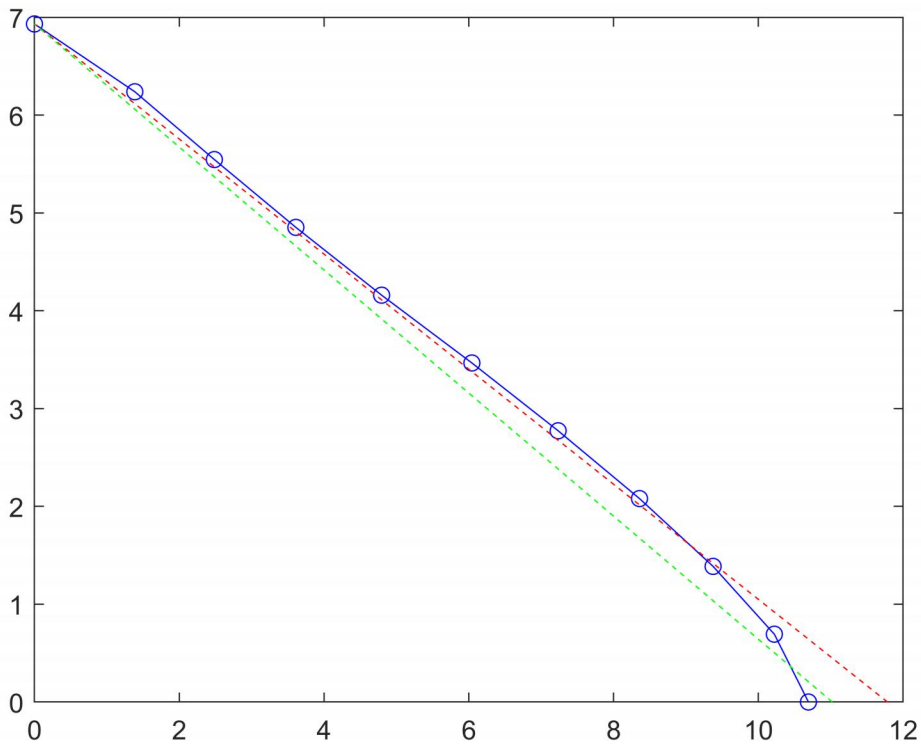

```
%(Do 1st-degree linear polynimial fit to data to find slope)
pf=polyfit(log(r'),log(n'),1)
```

```
pf = 1×2
   -1.5922   11.3545
```

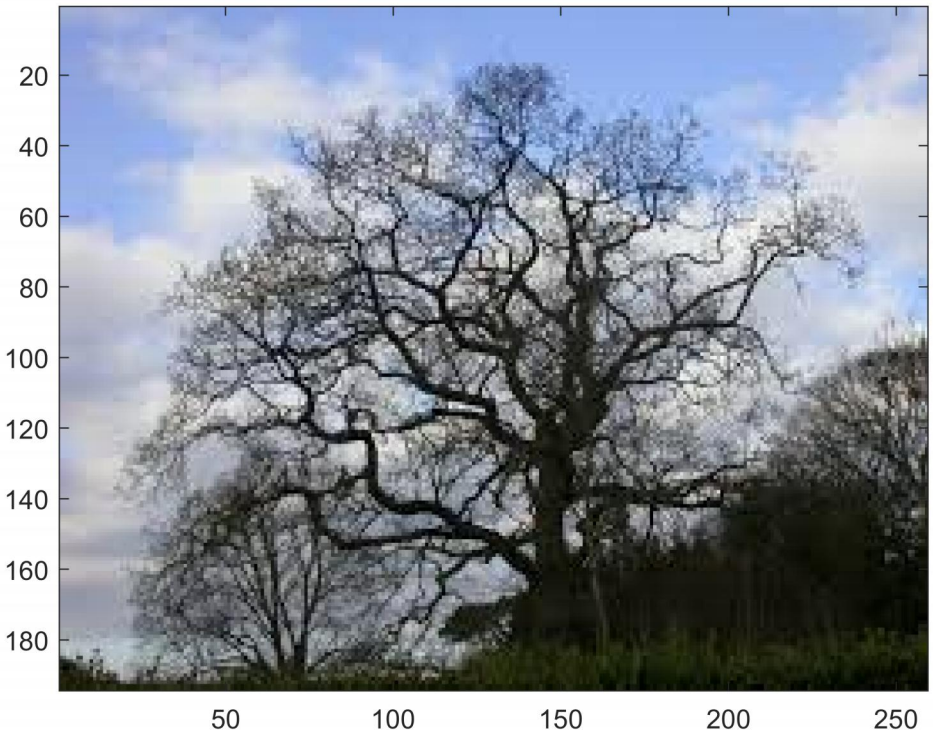
```
%(Look at local slope and notice region of stability.
boxcount(im1, 'slope')
```



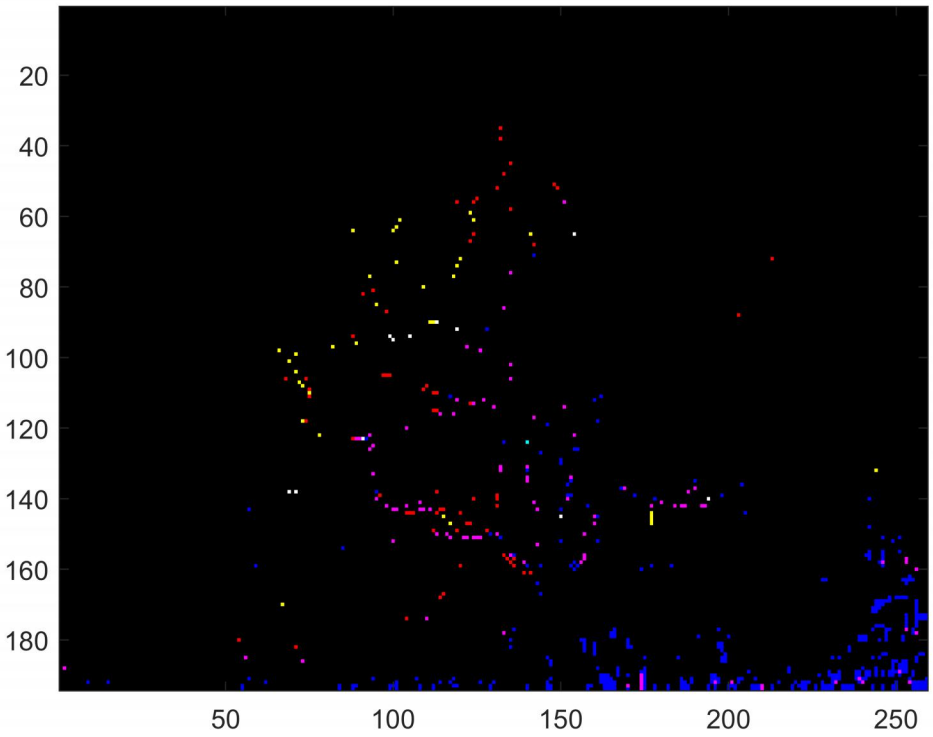

```
% (Compare data with lines from two different estimates of slope. Which is
% better and why?
plot(log(n),log(r),'bo-',log((r/r(end)).^(-1.7)), ...
    log(r),'r--',log((r/r(end)).^(-1.59)),log(r),'g--')
```



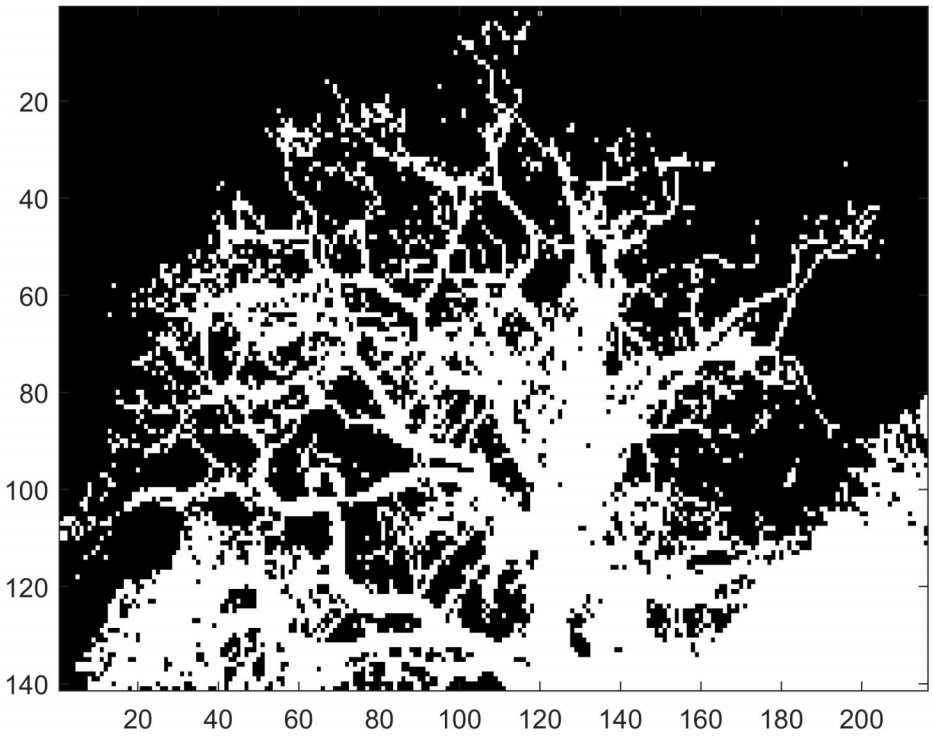

```
% Exercise 1
%(loads imag file)
im1 = imread('Tree1.jpg');
%(Look at image)
image(im1);
```



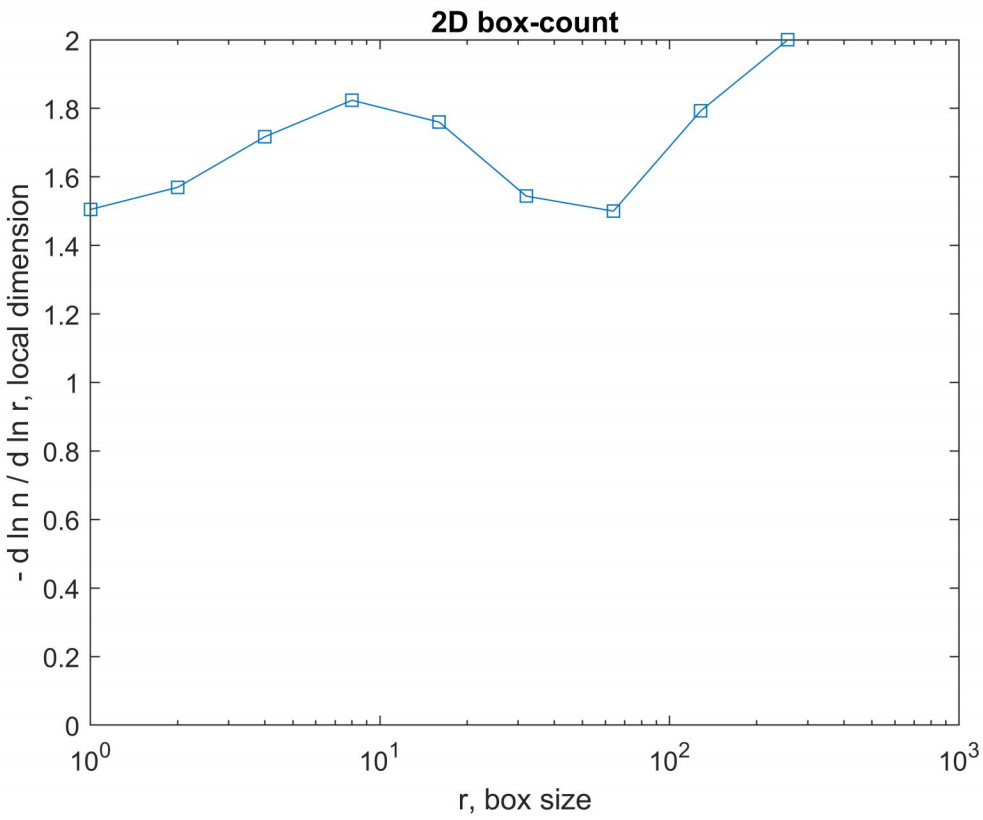

```
%(Only choose part of image with fractal structure you want to analyze)
i = im1(20:160, 25:240, 3);

%(Sets full color scale for image)
imagesc(~im1);

%(Converts to gray scale)
colormap gray;
```




```
%(Set threshhold to only look at data above half the maximum value)
th = (i<125);
%(Look at final image)
imagesc(th)
```




```
boxcount(th, 'slope')
```



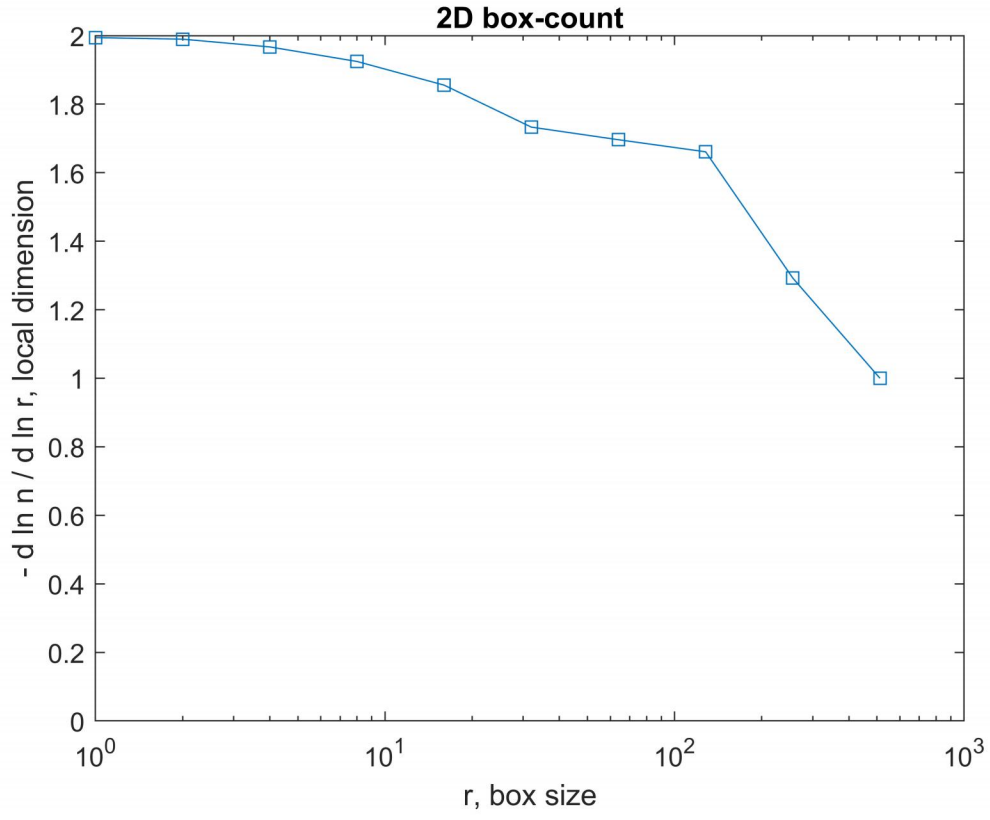
```
%(stores data in matrix for n=number of boxes, r=size/radius of boxes)
[n,r]=boxcount(th)
```

```
n = 1×9
       11259        3967        1278         367         102          32 ···
r = 1×9
     1     2     4     8    16    32    64   128   256
```

```
%(look at data, x axis is 1dt argument, y axis is 2nd argument)


%(Do 1st-degree linear polynimial fit to data to find slope)
pf=polyfit(log(r'),log(n'),1) % polyfit
```

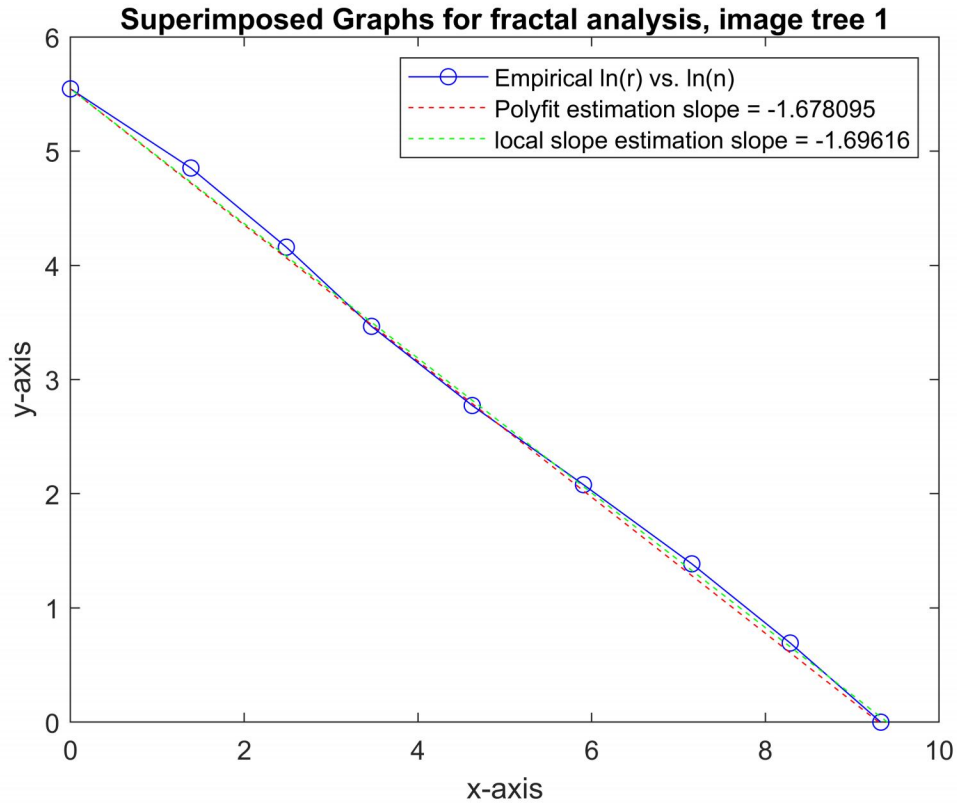
```
pf = 1×2
   -1.6781    9.3899
```

```
slope = pf(1);
%(Look at local slope and notice region of stability.

boxcount(im1, 'slope')
```




```
plot(log(n),log(r),'bo-',log((r/r(end)).^(-1.6781)), ...
    log(r),'r--',log((r/r(end)).^(-1.696)),log(r),'g--')
legend('Empirical ln(r) vs. ln(n)', ...
    sprintf(('Polyfit estimation slope = %f'), pf(1)),...
    ('local slope estimation slope = -1.69616'));
```

```
title('Superimposed Graphs for fractal analysis, image tree 1');
xlabel('x-axis');
ylabel('y-axis');
```




```
% 1B. question: How does the slope estimated from local slope analysis,
% i.e. boxcount(im1, 'slope'), compare
% with value calculated by the 1st-degree linear polynomial fit to data
% from the boxcount method?

% Answer:
% the polyfit estimation method slope is -1.678095
% the local slope estimation method slope is -1.69616

% 1C. question: Compare data with lines from two different estimates of slope.
% Which is better and why?
%
%
% Answer: In this question, we see whether the polyfit estimation or local slope
% estimation is closer to blue,
% to see which one is the better estimation. We can deduct that the local
% slope estimation is closer to the empirical ln(r) vs. ln(n) line.
% The fractal image analysis is better with local slope analysis because
% factal is made up of a matrix of numbers and the proportion stays the
% same even if the image portion is truncated. Whereas the polyfit line
% is done using regression, but the local slope estimation is done using true sample slope.
```

1 Ex.1 **40 / 40**

✓ **- 0 pts** *Correct*

- 10 pts Correctly submit greyscale image

- 10 pts Plot with 3 lines

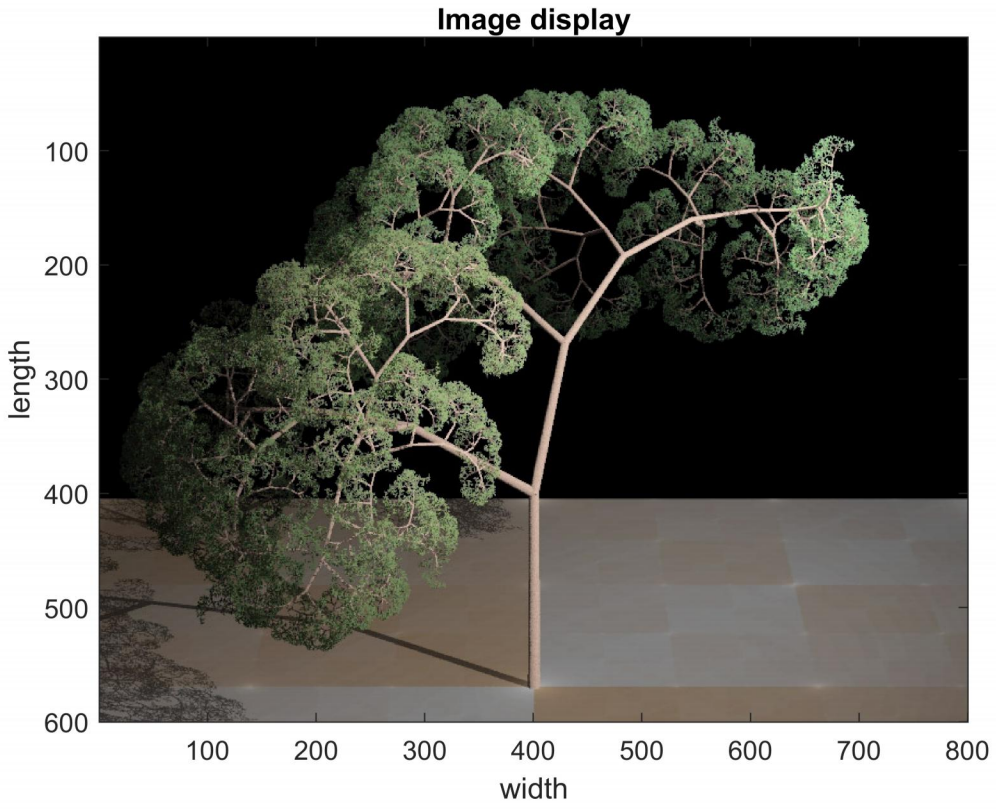
- 10 pts state that both are close to empirical solution

- 10 pts Correctly state that local slope analysis is better

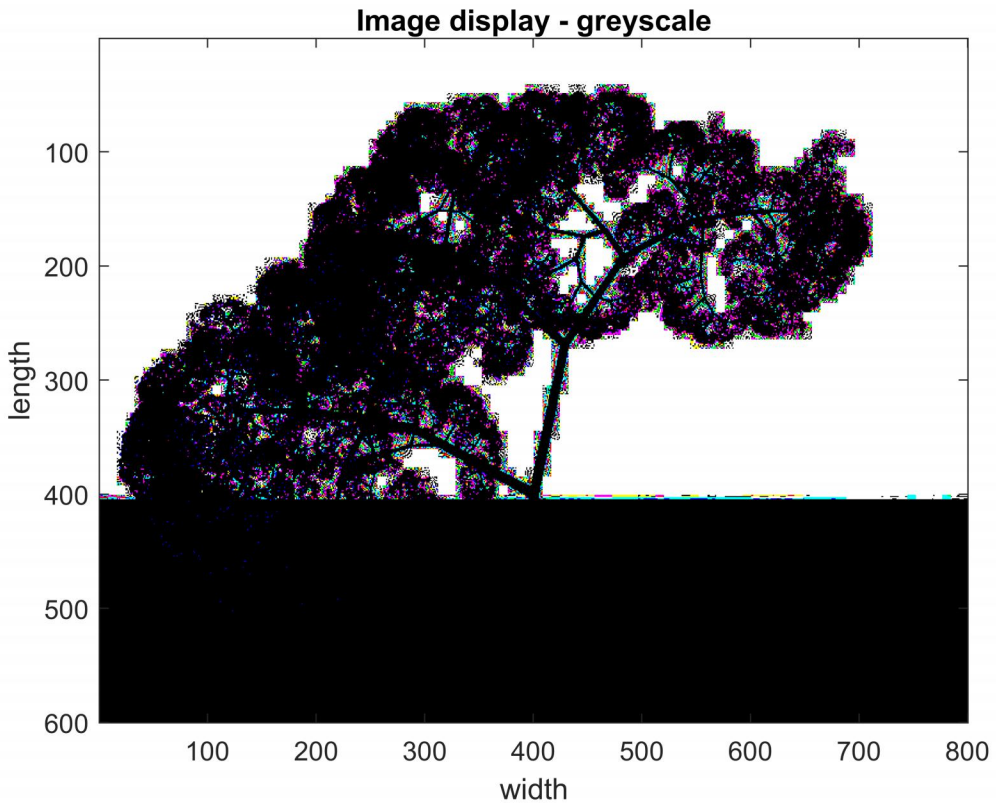
- 2 pts invalid explanation

```
%
```

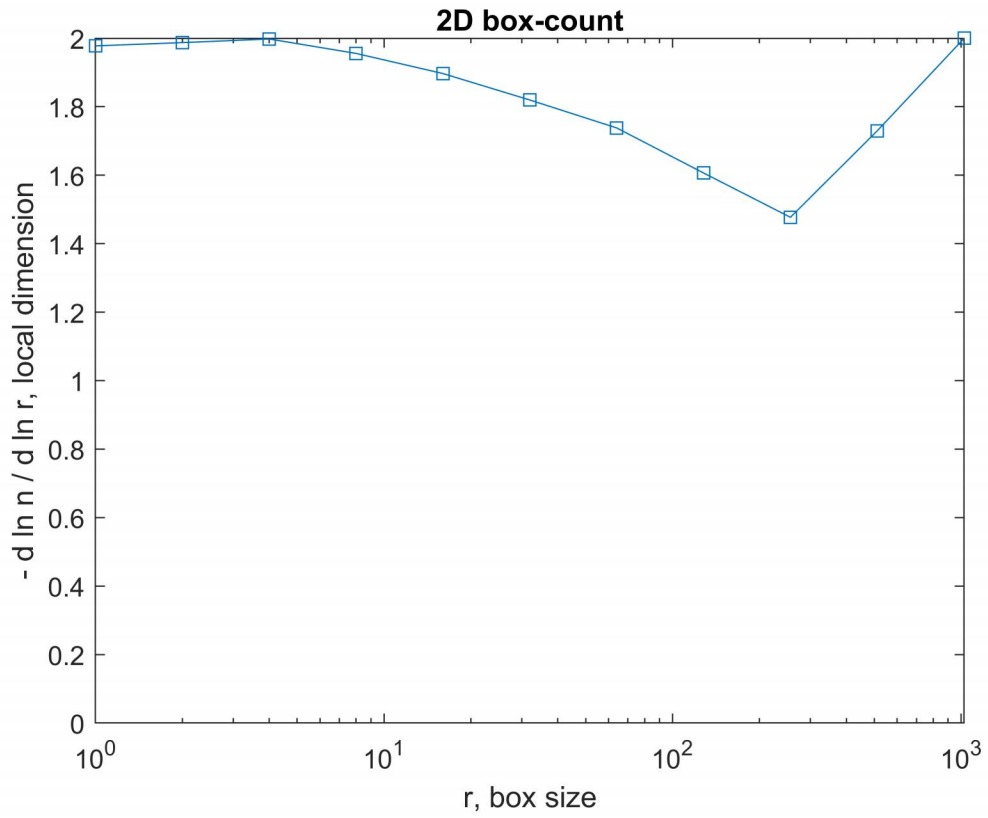
```
% Question 2:

%(loads imag file)
im1 = imread('Tree2.jpg');
%(Look at image)
image(im1);
xlabel('width');
ylabel('length');
title('Image display');
```




```

%(Sets full color scale for image)
imagesc(~im1);

%(Converts to gray scale)
colormap gray;
xlabel('width');
ylabel('length');
title('Image display - greyscale');
```




```
%(Set threshhold to only look at data above half the maximum value)
%(Look at final image)

boxcount(im1, 'slope')
```



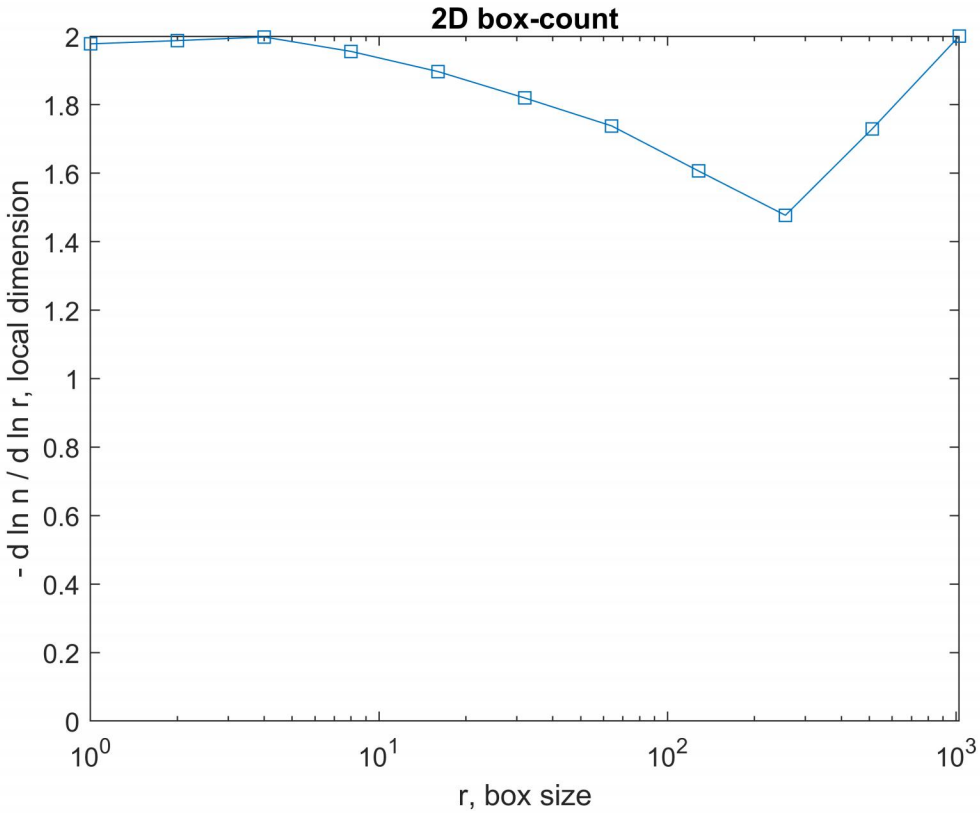

```
%(stores data in matrix for n=number of boxes, r=size/radius of boxes)
[n,r]=boxcount(im1)
```

```
n = 1×11
      300661       76332       19125        4785        1271         345 ···
r = 1×11
           1           2           4           8          16          32 ···
```

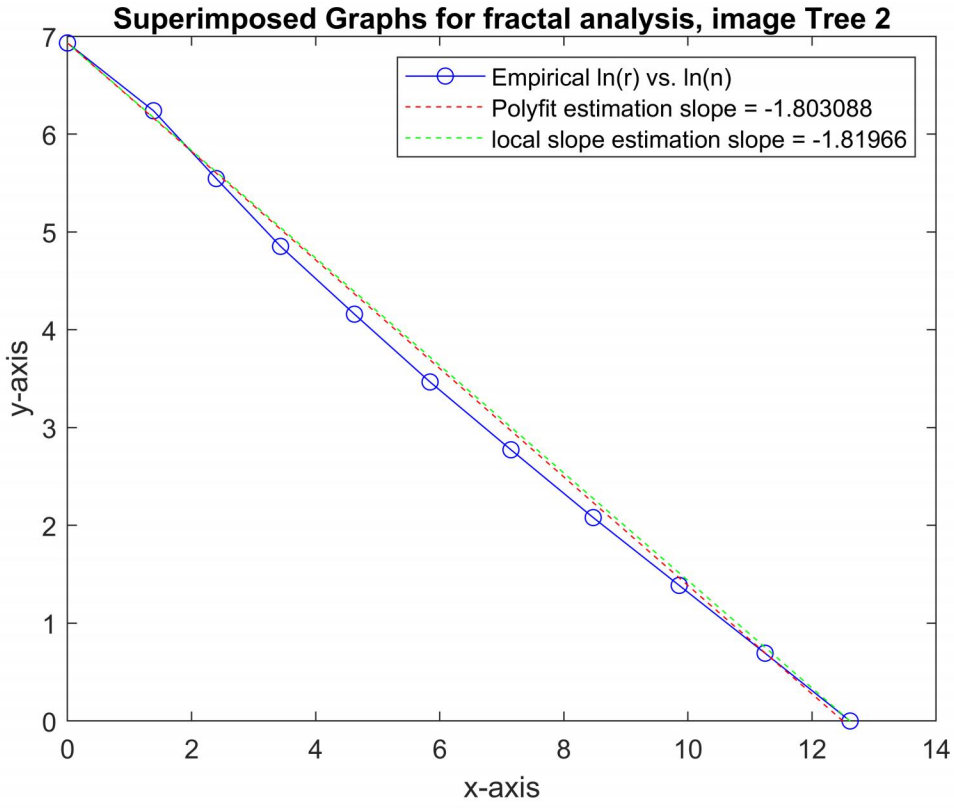
```
%(look at data, x axis is 1dt argument, y axis is 2nd argument)


%(Do 1st-degree linear polynimial fit to data to find slope)
pf=polyfit(log(r'),log(n'),1) % polyfit
```

```
pf = 1×2
   -1.8031   12.3420
```

```
%(Look at local slope and notice region of stability.

boxcount(im1, 'slope')
```




```
plot(log(n),log(r),'bo-',log((r/r(end)).^(-1.803088)), ...
    log(r),'r--',log((r/r(end)).^(-1.81966)),log(r),'g--')
legend('Empirical ln(r) vs. ln(n)', ...
    sprintf(('Polyfit estimation slope = %f'), pf(1)),...
    ('local slope estimation slope = -1.81966'));
title('Superimposed Graphs for fractal analysis, image Tree 2');
xlabel('x-axis');
ylabel('y-axis');
```




```
% Question 2B.
% How does the slope estimated from local slope analysis, i.e. boxcount(im1, %slope ), compare
% with value calculated by the 1st-degree linear polynomial fit to data from the boxcount metho

% Answer:
%
% the polyfit estimation method slope is -1.803088
% the local slope estimation method slope is -1.81966
%

% Question 2C.
% Compare data with lines from two different estimates of slope. Which is better and why?

% In this question, we see whether the polyfit estimation or local slope
% estimation is closer to empirical ln(r) vs. ln(n) curve.
% to see which one is the better estimation, we can deduct that the local
% slope estimation is closer to the empirical ln(r) vs. ln(n) line.
% The fractal image analysis is better with local slope analysis because
% factal is made up of a matrix of numbers and the proportion stays the
% same even if the image portion is truncated. Whereas the polyfit line
% is done using regression, but the local slope estimation is done using
% true sample slope.
```

2 Ex.2 **30 / 30**

✓ **- 0 pts** *Correct*

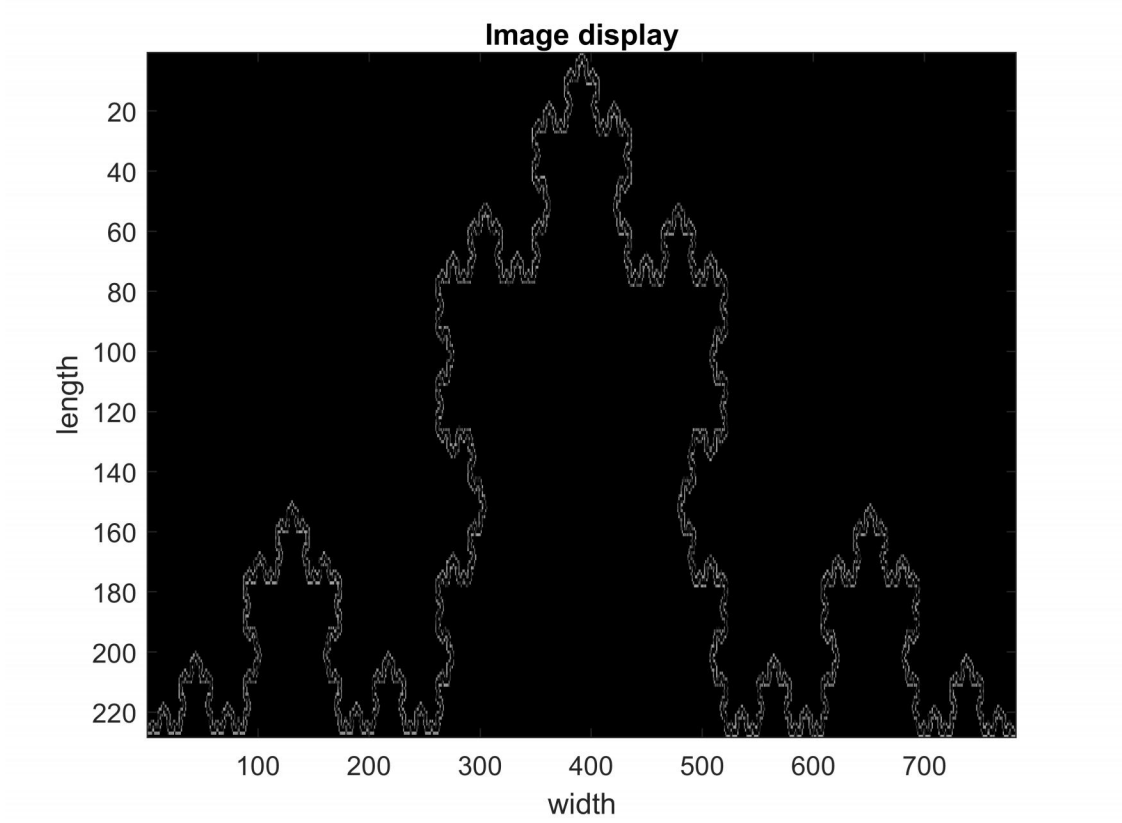
- 5 pts Greyscale image

- 3 pts Plot with 3 lines

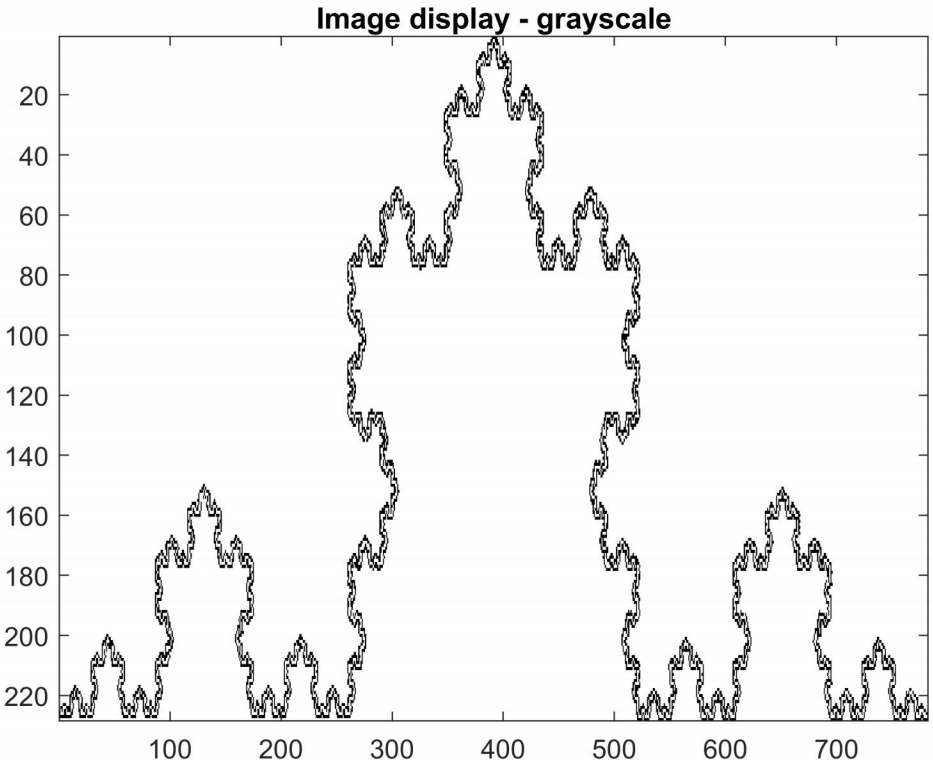
- 10 pts Both quite close to empirical solution

- 3 pts Local slope analysis is better + valid explanation

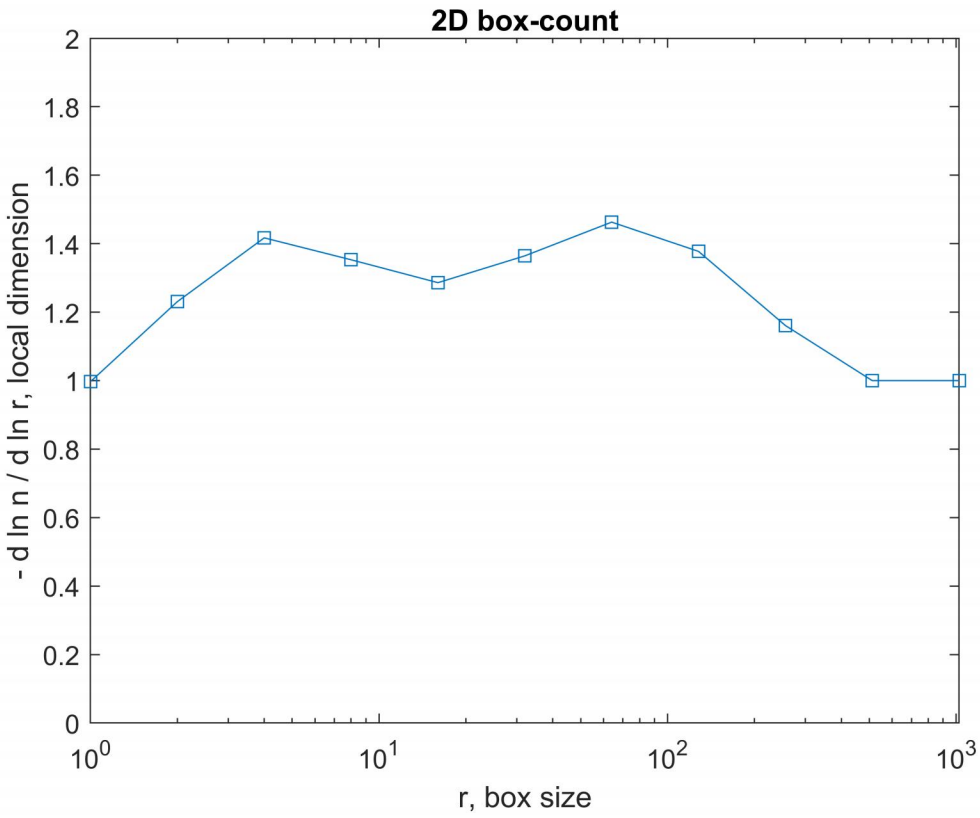
```
%(loads imag file)
im1 = imread('big_Koch.png');
%(Look at image)
image(im1);
%(Only choose part of image with fractal structure you want to analyze)
i = im1(20:160, 25:240, 3);
xlabel('width');
ylabel('length');
title('Image display');
```




```
%(Sets full color scale for image)
imagesc(~im1);


%(Converts to gray scale)
colormap gray;
title('Image display - grayscale');
```




```
%(Set threshhold to only look at data above half the maximum value)
%(Look at final image)

boxcount(im1, 'slope')
```



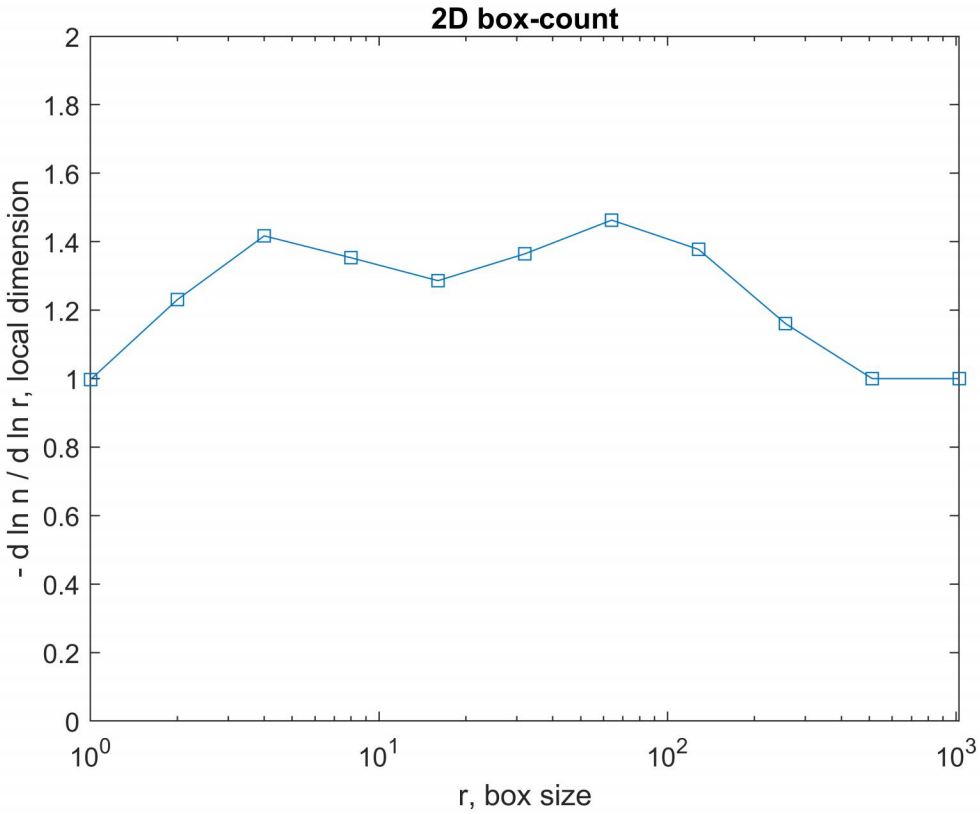

```
%(stores data in matrix for n=number of boxes, r=size/radius of boxes)
[n,r]=boxcount(im1)
```

```
n = 1×11
        6433        3223        1168         452         179          76 ···
r = 1×11
           1           2           4           8          16          32 ···
```

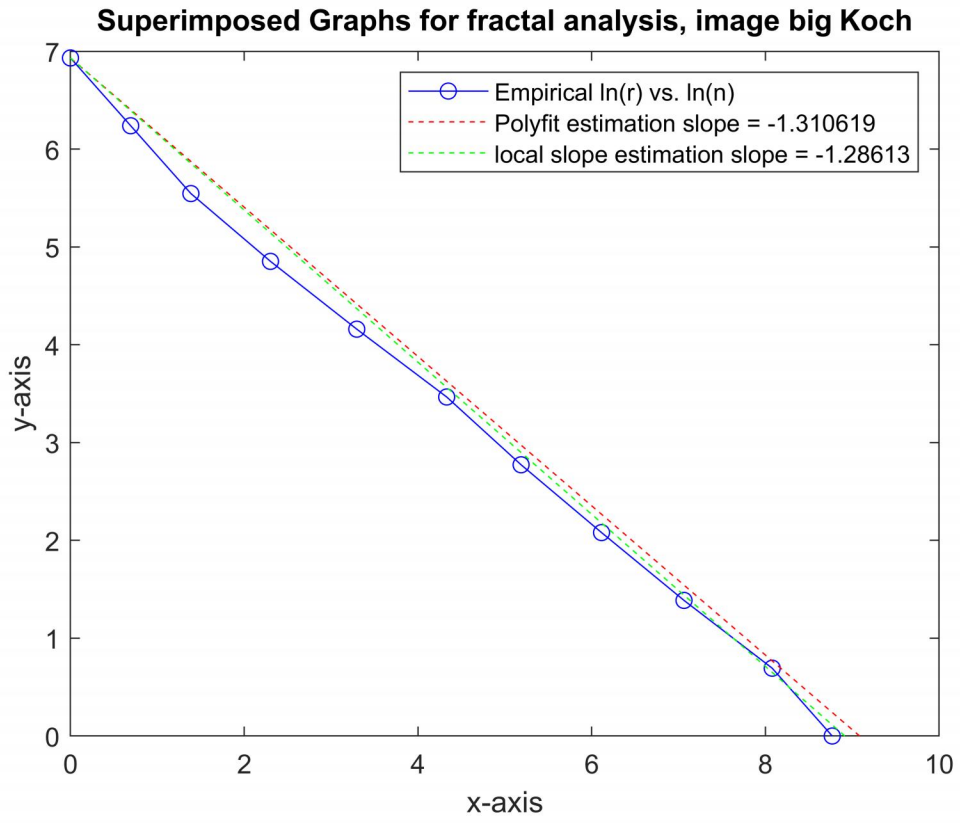
```
%(look at data, x axis is 1dt argument, y axis is 2nd argument)


%(Do 1st-degree linear polynimial fit to data to find slope)
pf=polyfit(log(r'),log(n'),1) % polyfit
```

```
pf = 1×2
   -1.3106    8.8350
```

```
%(Look at local slope and notice region of stability.

boxcount(im1, 'slope')
```




```
plot(log(n),log(r),'bo-',log((r/r(end)).^(-1.310619)), ...
    log(r),'r--',log((r/r(end)).^(-1.28613)),log(r),'g--')
legend('Empirical ln(r) vs. ln(n)', ...
    sprintf(('Polyfit estimation slope = %f'), pf(1)),...
    ('local slope estimation slope = -1.28613'));
title('Superimposed Graphs for fractal analysis, image big_ Koch');
xlabel('x-axis');
ylabel('y-axis');
```




```
% Question 3B.
%
% Answer:
%
% the polyfit estimation method slope is -1.310619
% the local slope estimation method slope is -1.28613
%

% Question 3C.
%
% Answer:
% In this question, we see whether the polyfit estimation or local slope
% estimation is closer to empirical ln(r) vs. ln(n) curve.
% To see which one is the better estimation, we can deduct that the local
% slope estimation is closer to the empirical ln(r) vs. ln(n) line.
% The fractal image analysis is better with local slope analysis because
% factal is made up of a matrix of numbers and the proportion stays the
% same even if the image portion is truncated. Whereas the polyfit line
% is done using regression, but the local slope estimation is done using
% true sample slope. It should not vary for a 1D image.
```

```
function [n,r] = boxcount(c,varargin)
%BOXCOUNT  Box-Counting of a D-dimensional array (with D=1,2,3).
%   [N, R] = BOXCOUNT(C), where C is a D-dimensional array (with D=1,2,3),
```

```
%   counts the number N of D-dimensional boxes of size R needed to cover
%   the nonzero elements of C. The box sizes are powers of two, i.e.,
%   R = 1, 2, 4 ... 2^P, where P is the smallest integer such that
%   MAX(SIZE(C)) <= 2^P. If the sizes of C over each dimension are smaller
%   than 2^P, C is padded with zeros to size 2^P over each dimension (e.g.,
%   a 320-by-200 image is padded to 512-by-512). The output vectors N and R
%   are of size P+1. For a RGB color image (m-by-n-by-3 array), a summation
%   over the 3 RGB planes is done first.
%
%   The Box-counting method is useful to determine fractal properties of a
%   1D segment, a 2D image or a 3D array. If C is a fractal set, with
%   fractal dimension DF < D, then N scales as R^(-DF). DF is known as the
%   Minkowski-Bouligand dimension, or Kolmogorov capacity, or Kolmogorov
%   dimension, or simply box-counting dimension.
%
%   BOXCOUNT(C,'plot') also shows the log-log plot of N as a function of R
%   (if no output argument, this option is selected by default).
%
%   BOXCOUNT(C,'slope') also shows the semi-log plot of the local slope
%   DF = - dlnN/dlnR as a function of R. If DF is contant in a certain
%   range of R, then DF is the fractal dimension of the set C. The
%   derivative is computed as a 2nd order finite difference (see GRADIENT).
%
%   The execution time depends on the sizes of C. It is fastest for powers
%   of two over each dimension.
%
%   Examples:
%
%      % Plots the box-count of a vector containing randomly-distributed
%      % 0 and 1. This set is not fractal: one has N = R^-2 at large R,
%      % and N = cste at small R.
%      c = (rand(1,2048)<0.2);
%      boxcount(c);
%
%      % Plots the box-count and the fractal dimension of a 2D fractal set
%      % of size 512^2 (obtained by RANDCANTOR), with fractal dimension
%      % DF = 2 + log(P) / log(2) = 1.68 (with P=0.8).
%      c = randcantor(0.8, 512, 2);
%      boxcount(c);
%      figure, boxcount(c, 'slope');
%
%   F. Moisy
%   Revision: 2.10,  Date: 2008/07/09


% History:
% 2006/11/22: v2.00, joined into a single file boxcountn (n=1,2,3).
% 2008/07/09: v2.10, minor improvements

% control input argument
```

```
error(nargchk(1,2,nargin));

% check for true color image (m-by-n-by-3 array)
if ndims(c)==3
    if size(c,3)==3 && size(c,1)>=8 && size(c,2)>=8
        c = sum(c,3);
    end
end

warning off
c = logical(squeeze(c));
warning on

dim = ndims(c); % dim is 2 for a vector or a matrix, 3 for a cube
if dim>3
    error('Maximum dimension is 3.');
end

% transpose the vector to a 1-by-n vector
if length(c)==numel(c)
    dim=1;
    if size(c,1)~=1
        c = c';
    end
end

width = max(size(c));    % largest size of the box
p = log(width)/log(2);   % nbre of generations

% remap the array if the sizes are not all equal,
% or if they are not power of two
% (this slows down the computation!)
if p~=round(p) || any(size(c)~=width)
    p = ceil(p);
    width = 2^p;
    switch dim
        case 1
            mz = zeros(1,width);
            mz(1:length(c)) = c;
            c = mz;
        case 2
            mz = zeros(width, width);
            mz(1:size(c,1), 1:size(c,2)) = c;
            c = mz;
        case 3
            mz = zeros(width, width, width);
            mz(1:size(c,1), 1:size(c,2), 1:size(c,3)) = c;
            c = mz;
    end
end
```

```
n=zeros(1,p+1); % pre-allocate the number of box of size r

switch dim

    case 1        %------------------- 1D boxcount --------------------%

        n(p+1) = sum(c);
        for g=(p-1):-1:0
            siz = 2^(p-g);
            siz2 = round(siz/2);
            for i=1:siz:(width-siz+1)
                c(i) = ( c(i) || c(i+siz2));
            end
            n(g+1) = sum(c(1:siz:(width-siz+1)));
        end

    case 2         %------------------- 2D boxcount --------------------%

        n(p+1) = sum(c(:));
        for g=(p-1):-1:0
            siz = 2^(p-g);
            siz2 = round(siz/2);
            for i=1:siz:(width-siz+1)
                for j=1:siz:(width-siz+1)
                    c(i,j) = ( c(i,j) || c(i+siz2,j) || c(i,j+siz2) || c(i+siz2,j+siz2) );
                end
            end
            n(g+1) = sum(sum(c(1:siz:(width-siz+1),1:siz:(width-siz+1))));
        end

    case 3         %------------------- 3D boxcount --------------------%

        n(p+1) = sum(c(:));
        for g=(p-1):-1:0
            siz = 2^(p-g);
            siz2 = round(siz/2);
            for i=1:siz:(width-siz+1),
                for j=1:siz:(width-siz+1),
                    for k=1:siz:(width-siz+1),
                        c(i,j,k)=( c(i,j,k) || c(i+siz2,j,k) || c(i,j+siz2,k) ...
                            || c(i+siz2,j+siz2,k) || c(i,j,k+siz2) || c(i+siz2,j,k+siz2) ...
                            || c(i,j+siz2,k+siz2) || c(i+siz2,j+siz2,k+siz2));
                    end
                end
            end
            n(g+1) = sum(sum(sum(c(1:siz:(width-siz+1),1:siz:(width-siz+1),1:siz:(width-siz+1)
        end

end
```

```
n = n(end:-1:1);
r = 2.^(0:p); % box size (1, 2, 4, 8...)

if any(strncmpi(varargin,'slope',1))
    s=-gradient(log(n))./gradient(log(r));
    semilogx(r, s, 's-');
    ylim([0 dim]);
    xlabel('r, box size'); ylabel('- d ln n / d ln r, local dimension');
    title([num2str(dim) 'D box-count']);
elseif nargout==0 || any(strncmpi(varargin,'plot',1))
    loglog(r,n,'s-');
    xlabel('r, box size'); ylabel('n(r), number of boxes');
    title([num2str(dim) 'D box-count']);
end
if nargout==0
    clear r n
end
end
```

3 Ex.3 **30 / 30**

✓ **- 0 pts** *Correct*

- 5 pts Grayscale image

- 3 pts Plot with 3 lines

- 10 pts both close to empirical solution

- 5 pts Local slope better or sufficient explanation