Math 498
Homework #2

Chartier/Greenbaum
Winter 2008

Homework due – Friday, January 25, 2008

1. The following exercise can be computed by hand as opposed to using MATLAB.

   Let us form a Vector Space Model using the following dictionary of keywords:

   chocolate, ice cream, sprinkles

   Suppose our list of documents is:

   D1. *I eat only the chocolate icing off the cake*

   D2. *I like chocolate and vanilla ice cream*

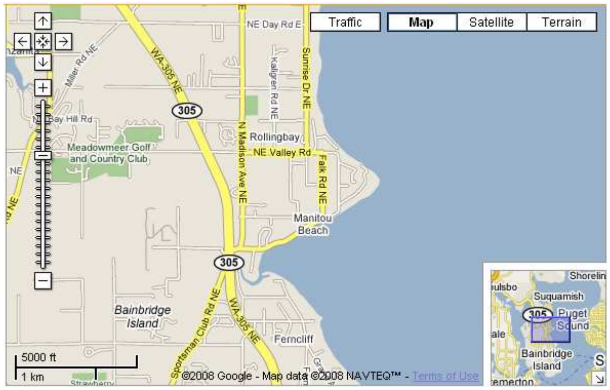
   D3. *Children like chocolate cake with sprinkles*

   D4. *Can I have another scoop of ice cream if you hold both the sprinkles and chocolate sauce*

   (a) Form the document matrix A with normalized columns.

   (b) Using the Vector Space Model, as outlined in class, rank the documents D1, D2, D3, and D4 according to their computed relevance to the query: `chocolate, ice cream`.

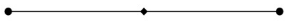
2. In this exercise, we create a fractal coastline. Below we see a portion of the coastline of Bainbridge Island as depicted on Google Maps.



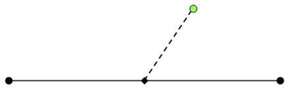
   We will use an iterative algorithm similar to that described in class and the course notes to create 2D fractal landscapes. The algorithm is:

   0. Begin with one straight line.

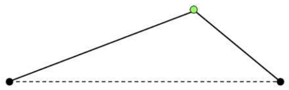
   1. For each line in the current curve, find the midpoint, denoted by a solid diamond in the picture below.



   2. Create a new point by moving a random amount in the x and y directions from that midpoint as seen below.

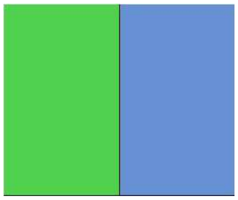


3. Connect the endpoints of the original line with the new point.

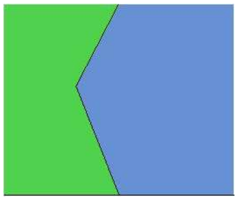


4. If desired number of total iterates completed then stop, else go to Step 1.

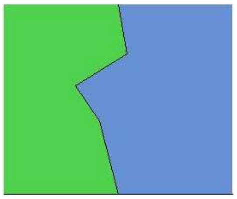
Note we have not specified the amount of reduction per iterate that reduces the size of the interval in which the random numbers are selected for the amount of displacement in the x and y directions in Step 2. You will determine a suitable size of reduction in your work. One such implementation created the figures below.



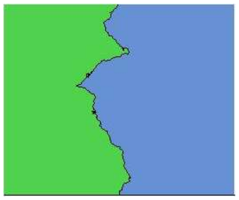
Iterate 0



Iterate 1



Iterate 2

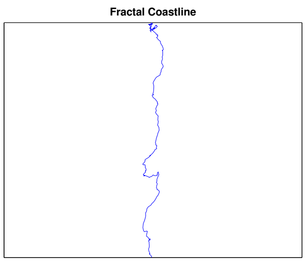


Iterate 8

Further, plotting the fractal coastline will take a different command than that used for the 2D fractal landscapes since we will not have uniform spacing between x values. Assuming your implementation stores the x values and y values for points on the fractal coastline in the vectors `xValues` and `yValues`, respectively. Then the MATLAB commands:

```
plot(xValues,yValues)
axis equal
```

will plot the fractal coastline with a 1:1 aspect ratio for the axes. Note that while the example graphs given above color the land and water masses, your code does not need to do this. (In fact, the commands above will not do this although you may wish to explore the `fill` command if you desire such coloring.) Your implementation need only to plot the curve which is the coastline created by your algorithm as seen below.



For this problem submit a listing of your MATLAB code that constructs such fractal coastlines and a write-up which describes how your implementation of this algorithm reduces the size of the interval in which the random numbers are selected for the amount of displacement in the x and y directions in Step 2. Your write-up should also include at least two fractal coastlines produced by your MATLAB code.

3. Suppose you have a pseudorandom number generator (called `rand`) that generates random numbers uniformly distributed between 0 and 1. How could you use it to generate a discrete random variable that takes on the values 1 through 6, each with equal probability?

4. Write a MATLAB code to simulate rolling of a fair die; that is, it should generate a discrete random variable that takes on the values 1 through 6, each with probability 1/6. Run the code 1000 times and plot a bar graph showing the number of 1's, 2's, 3's, etc. Your code might look something like the following:

```
bins = zeros(6,1);
for count=1:1000,
%   Generate a random number k with value 1,2,3,4,5, or 6, each with prob. 1/6.
%   (You fill in this section.)

  bins(k) = bins(k) + 1;        % Count the times each number is generated.
end;
bar(bins)                       % Plot the bins.
```

Turn in your plot and a listing of your code. Determine the expected value and the variance of the random variable k.

Now let A_{1000} be the average value generated. Run the code 100 times, each time recording the average value A_{1000}. For example, you might modify the above code as follows:

```
A = zeros(100,1);                % Initialize average values.
for n_tries=1:100,               % Try the experiment 100 times.

  for count=1:1000,
%     Generate a random number k with value 1,2,3,4,5, or 6, each with prob. 1/6.
%     (You fill in this section.)

    A(n_tries) = A(n_tries) + k; % Add each value k.
  end;
  A(n_tries) = A(n_tries)/1000;  % Divide by 1000 to get the average of k.

end;
hist(A)                          % This plots a histogram of average values.
```

Turn in your histogram and a listing of your code. How does the random variable A_{1000} appear to be distributed (uniform, normal, or some other distribution)? Approximately what is its mean, its variance, and its standard deviation? (Hint: Use the Central Limit Theorem.)