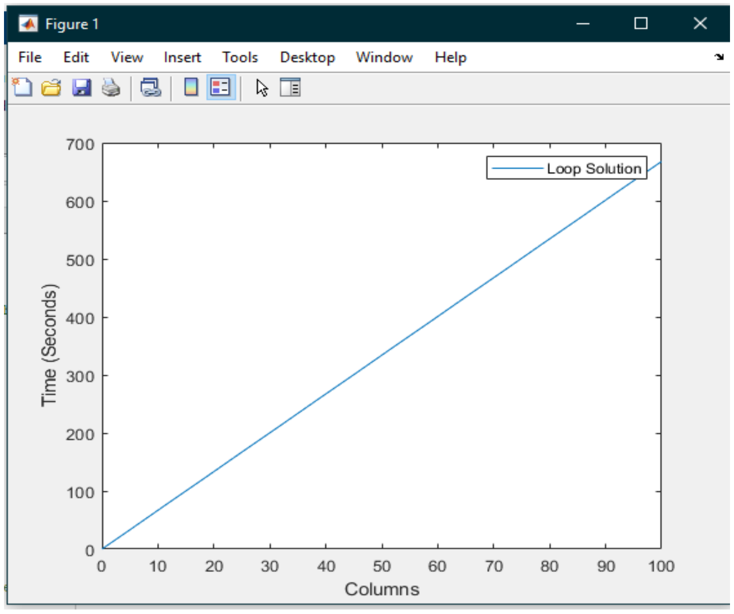
James Murphy
CS 381 - Image Processing
Homework #1

Problem 1 : Part 1

For part Problem 1 Part 1 we're finding the euclidian distance between 2 matrices. For my implementation I took 2 matrices and randomly generated entries in a 100 x 100 matrix. To find the Euclidian distance you have to take the sum of the row from i to column n and subtract it from the row of matrix_b using the j loop as the row variable. That way it's mat_A(row1, n) - mat_B(row1, n) , mat_A(row1, col) - mat_B(row2, col) …. mat_A(row2, col) - mat_B(row1, col) … mat_A(i, col) - mat_B(j, col). Sum those values together, square it and then take the square root of it and then store those values in the i, j pair of a new matrix c.

```
for i = 1 : m
    for j = 1 : n

        %to find the i, j entry of the euclidian distance you take the row
        %of matrix a from i to m and subtract it
        %from the row of matrix b from j to m and then square it, sum it and then
        %take the square root of it according to the formula given
        mat_c(i, j) = sqrt(sum((mat_a(i, :) - mat_b(j, :)).^2));

        %increment j
        j = j + 1;
    end
    %increment i
    i = i + 1;
end
```

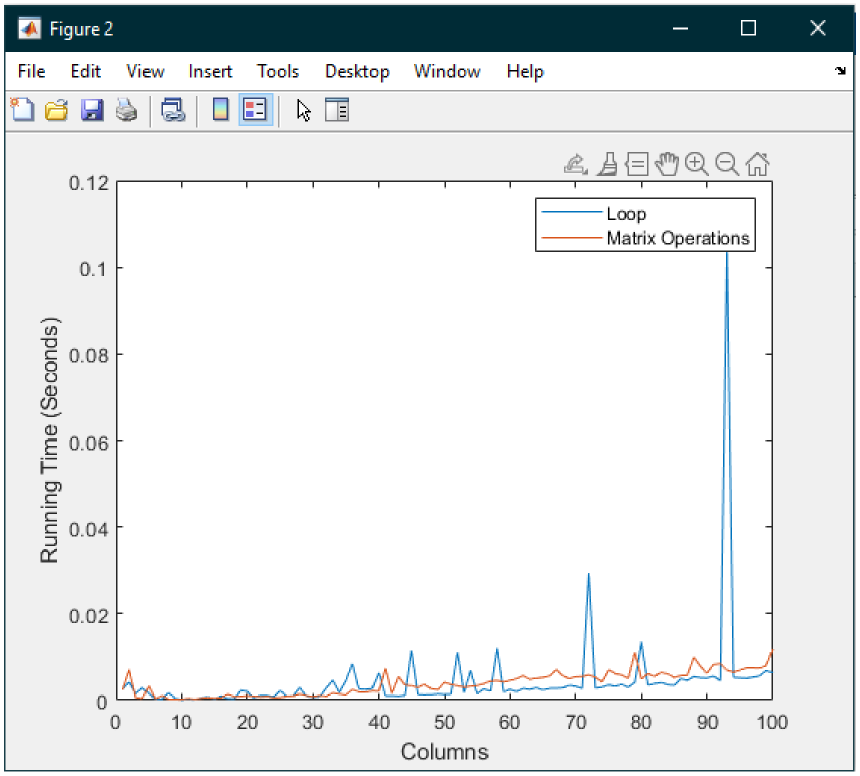
Graph of Part 1 :



Problem 1 : Part 2 : We take the randomly generated matrices a and b and subtract them and store it in d. Then we sum the values of matrix d together and store into a new matrix euc_dist. Take the euc_d matrix, square it and then sum it.

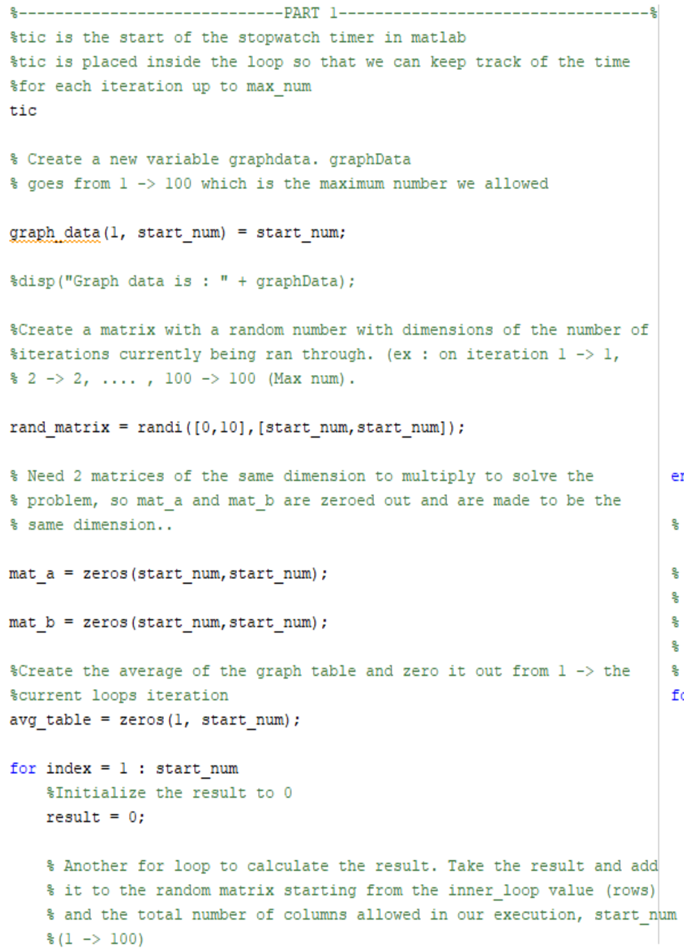
```
% Part 2

tic;

% Initalize a variable matrix d and subtract a and b and store it in d.
mat_d = mat_a - mat_b;
% take the sum of the matrix
euc_dist = sum(mat_d);

% square the matrix
euc_dist = (euc_dist .^2);

% take the square root of the matrix to get the euclidian distance
euc_dist = sqrt(euc_dist);

end_time_pt2 = toc;
```

Graph Problem 1 Part 2 :

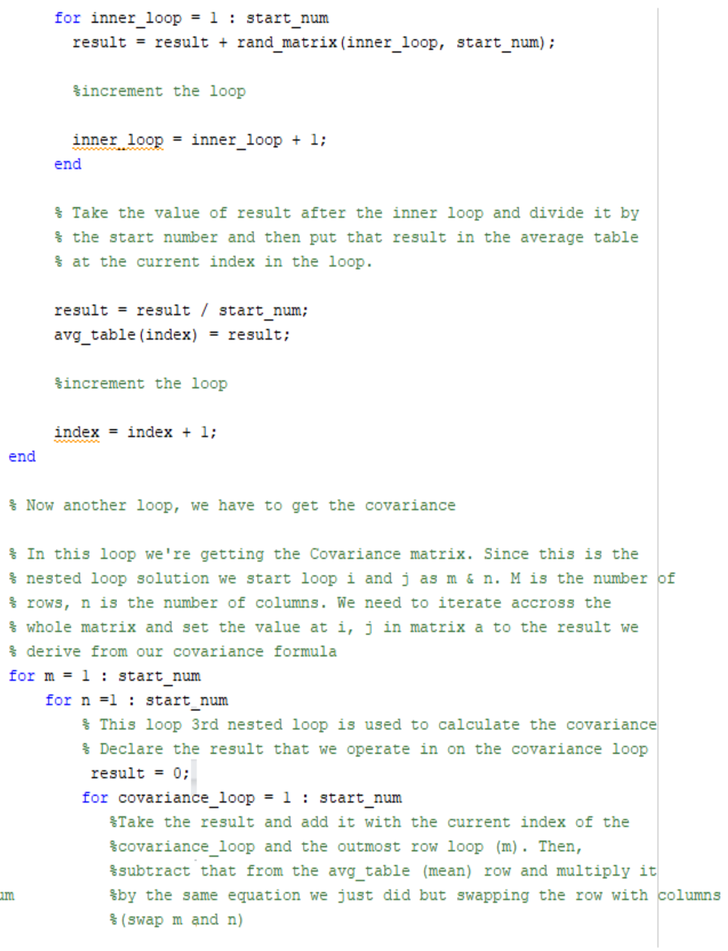


Problem 2 : Part 1 :

Sorry for the long code, it can’t fit in one image and I commented on it heavily. We create a variable max_num and set it to 100. Then create a random matrix with values. We need 2 matrices of the same dimension so we zero out a matrix of size 100 x 100. We then need to find the mean / average. So a table is created for it and it’s zeroed out from 1 -> 100. Then a loop is created to calculate the result. The result is taken and added with itself and the random matrix from 1 -> n which is 100. Then we take the result and divide it by 100 since we did 100 iterations and set the current index in the loop to that value. After getting the average table we can now find the covariance. We initialize another result variable and take the result and add it with the (random_matrix(loop index, and the row index) - the average in our table at m * the random_matrix(loop index, and the column index - the average in our table at n). Then we store that result into the i, j pair of the matrix. After that we need to calculate the standard deviation. So we take the diagonal of the matrix we just stored the i, j pairs in transpose it and square root that matrix. Then we take matrix b and divide both matrix a and b at m, n in the loop by the standard deviation at m and n respectively.



```
%----------------------------PART 1---------------------------------%
%tic is the start of the stopwatch timer in matlab
%tic is placed inside the loop so that we can keep track of the time
%for each iteration up to max_num
tic

% Create a new variable graphdata. graphData
% goes from 1 -> 100 which is the maximum number we allowed

graph_data(1, start_num) = start_num;

%disp("Graph data is : " + graphData);

%Create a matrix with a random number with dimensions of the number of
%iterations currently being ran through. (ex : on iteration 1 -> 1,
% 2 -> 2, .... , 100 -> 100 (Max num).

rand_matrix = randi([0,10],[start_num,start_num]);

% Need 2 matrices of the same dimension to multiply to solve the
% problem, so mat_a and mat_b are zeroed out and are made to be the
% same dimension..

mat_a = zeros(start_num,start_num);

mat_b = zeros(start_num,start_num);

%Create the average of the graph table and zero it out from 1 -> the
%current loops iteration
avg_table = zeros(1, start_num);

for index = 1 : start_num
    %Initialize the result to 0
    result = 0;

    % Another for loop to calculate the result. Take the result and add
    % it to the random matrix starting from the inner_loop value (rows)
    % and the total number of columns allowed in our execution, start_num
    %(1 -> 100)
```

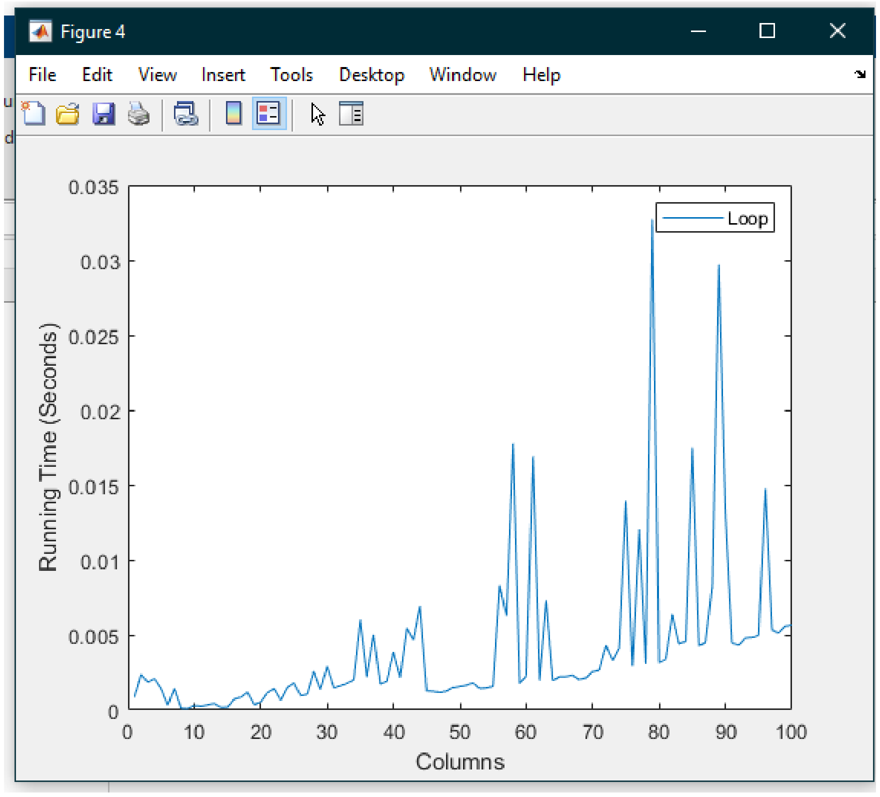


```
     for inner_loop = 1 : start_num
       result = result + rand_matrix(inner_loop, start_num);

       %increment the loop

       inner_loop = inner_loop + 1;
     end

     % Take the value of result after the inner loop and divide it by
     % the start number and then put that result in the average table
     % at the current index in the loop.

     result = result / start_num;
     avg_table(index) = result;

     %increment the loop

     index = index + 1;
end

% Now another loop, we have to get the covariance

% In this loop we're getting the Covariance matrix. Since this is the
% nested loop solution we start loop i and j as m & n. M is the number of
% rows, n is the number of columns. We need to iterate accross the
% whole matrix and set the value at i, j in matrix a to the result we
% derive from our covariance formula
for m = 1 : start_num
    for n =1 : start_num
        % This loop 3rd nested loop is used to calculate the covariance
        % Declare the result that we operate in on the covariance loop
         result = 0;
        for covariance_loop = 1 : start_num
           %Take the result and add it with the current index of the
           %covariance_loop and the outmost row loop (m). Then,
           %subtract that from the avg_table (mean) row and multiply it
           %by the same equation we just did but swapping the row with columns
           %(swap m and n)
```

```
            result = result + (rand_matrix(covariance_loop, m)
            - avg_table(m) * (rand_matrix(covariance_loop, n) - avg_table(n)));

            %Then, store that result into the i, j pair of our current
            %matrix.
            mat_a(m, n) = result;

            %increment the loop
            covariance_loop = covariance_loop + 1;

        end

        %increment the loop
        n = n + 1;
    end

    %increment the loop
    m = m +1;
end

%Now we need to calculate the standard deviation

%Take the diagonal of our matrix
standard_deviation = diag(mat_a);

%Tranpose the diagonal of the matrix
standard_deviation = transpose(standard_deviation);

%Take the square root of the transpose of the diagonal
standard_deviation = sqrt(standard_deviation);

%M = rows, N = cols

for m = 1:start_num
    for n = 1:start_num

        %We take the zerod out matrix b, and take the result of the i,
        %j pair of covariance matrix i and divide it by the standard
        %deviation using the rows variable m

        mat_b(m,n) = mat_a(m, n) / standard_deviation(m);
```

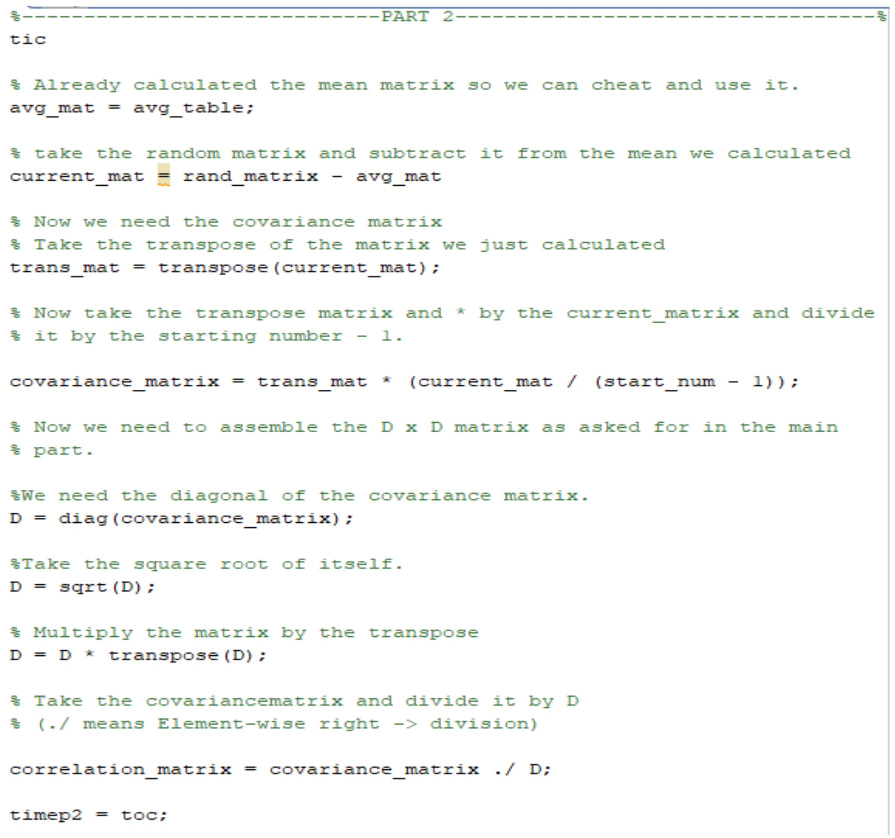
```
%Take the square root of the transpose of the diagonal
standard_deviation = sqrt(standard_deviation);

%M = rows, N = cols

for m = 1:start_num
    for n = 1:start_num

        %We take the zerod out matrix b, and take the result of the i,
        %j pair of covariance matrix i and divide it by the standard
        %deviation using the rows variable m

        mat_b(m,n) = mat_a(m, n) / standard_deviation(m);

        % After that we take the value we jsut stored at the i, j pair
        %and divide it by the standard deviation.

        mat_b(m,n) = mat_b(m, n) / standard_deviation(n);

        %increment the loop
        n = n + 1;
    end

    %increment the loop
    m = m + 1;

end

% mat_b is now the correlation matrix

% Toc stops the timer, store it in variable end_time so we can update
% the graph on the elapsed time.

timep1 = toc;
```

Graph of Problem 2 Part 1 :



Problem 2 Part 2 :

In part 2 of the problem, I was able to ‘cheat’ because we already had the mean / average matrix calculated from the first part of the problem so I used that. I created a new matrix called “current_mat” and subtracted it from the rand_matrix previously created from the first part of this problem. I took the transpose of current_matrix. To calculate the covariance matrix we needed to take the transpose matrix and multiply it by the current_matrix / the iteration in the “overall” loop the whole problem is in - 1. Next to calculate D we take the diagonal of the covariance matrix, square root it and then multiply it by the transpose of D. After that to calculate the correlation matrix we take the covariance matrix and divide it by D.



```
%-----------------------------PART 2-----------------------------------%
tic

% Already calculated the mean matrix so we can cheat and use it.
avg_mat = avg_table;

% take the random matrix and subtract it from the mean we calculated
current_mat = rand_matrix - avg_mat

% Now we need the covariance matrix
% Take the transpose of the matrix we just calculated
trans_mat = transpose(current_mat);

% Now take the transpose matrix and * by the current_matrix and divide
% it by the starting number - 1.

covariance_matrix = trans_mat * (current_mat / (start_num - 1));

% Now we need to assemble the D x D matrix as asked for in the main
% part.

%We need the diagonal of the covariance matrix.
D = diag(covariance_matrix);

%Take the square root of itself.
D = sqrt(D);

% Multiply the matrix by the transpose
D = D * transpose(D);

% Take the covariancematrix and divide it by D
% (./ means Element-wise right -> division)

correlation_matrix = covariance_matrix ./ D;

timep2 = toc;
```

Graph of Problem 2 Part 2 :

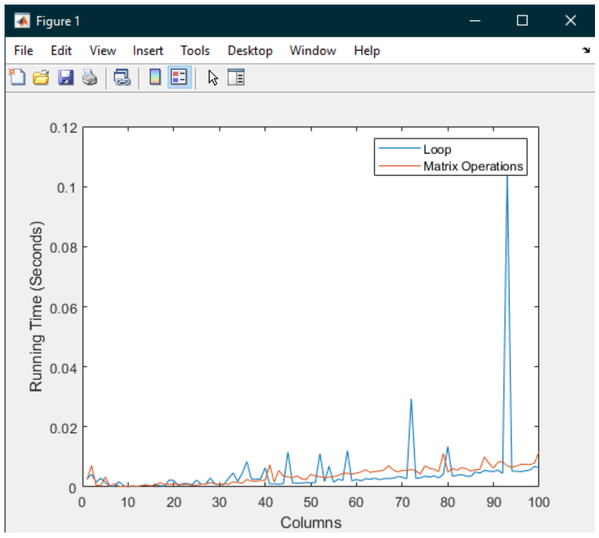


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Loop
Matrix Operations
Running Time (Seconds)
Columns