William O’Mullan

ECE 3522: Stochastic Processing

Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 1912

# Problem Statement

This assignment focuses on the concept of covariance and how we can visualize it as a probability density function of two random variables. We are required to evaluate the three dimensional visualizations of both uniform and Gaussian distributions in two separate tasks. In the first task we are simply to find the joint probability distribution of two uniformly distributed random variables in the range . The next task is slightly more complicated, we are to take a set of four covariance matrices and generate two normally distributed random vectors that fulfill the requirements set by the covariance matrices for the means . Once we have these Gaussian random vectors, we are to find their joint probability distribution in the same fashion as we did for the uniform random vectors.

# Approach and Results

The joint probability distribution of two random variables, and , is simply their product if both vectors are independent of each other. In this experiment we choose for and to be independent for both tasks.

For discrete random vectors, this equates to the dot product between the PDF of one vector and the transpose of the PDF of the other, depending on whether they are column or row vectors the order changes. Since I like to use column vectors in my code my joint probability distribution could be found by:

This is enough to complete our first task, but for the second a solid understanding of cross-covariance is required. The cross-covariance for two discrete random variables is:

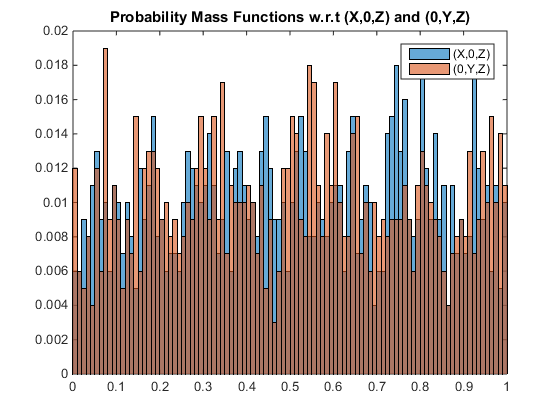
This gives us a covariance matrix between the random variables and . Along the main diagonal of the covariance matrix are the variances of each variable and on the reverse diagonal are the covariances of each variable. For our two by two matrices we are given in lab this means that:

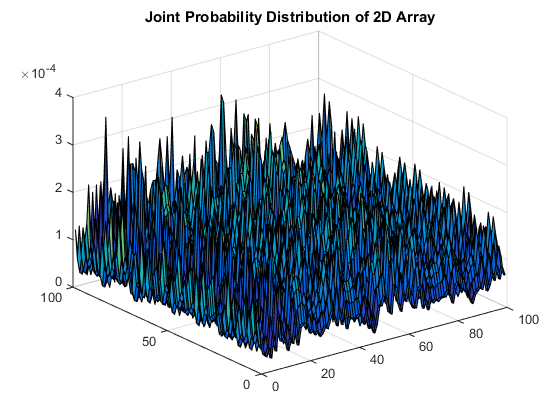
Luckily for us, MATLAB contains a function *mvnrnd()* that can take the argument of two means, a covariance matrix and desired vector size to generate two normally distributed random vectors who follow the parameters we specify. If one was not quite sensible enough, he might try to code that function from scratch and end up so far down a rabbit hole made of linear algebra that he postpones finishing the assignment out of desperate foolishness.

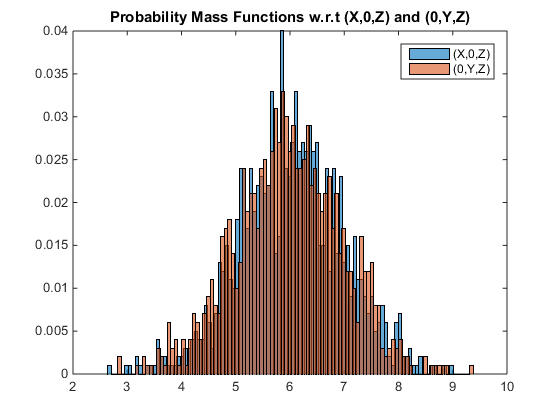
# MATLAB Code

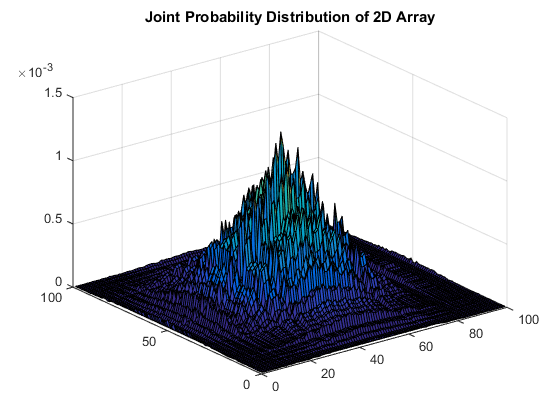
% Computer Assignment no. 7  
% Stochastic Processing  
% Bill O'Mullan  
  
function Stochastics\_ca\_7  
clear;close all;clc;  
  
% Establish counter for how many times plot\_3Djpd() is called  
count1 = 0;  
count2 = 1;  
  
% Generate a 2D array of random numbers between 0 and 1  
rand\_num = generate\_2Drand('Uniform',1e3);  
  
% Plot the 3D Probability Density Function of the random numbers generated  
count1 = (2\*count1) + 1;  
count2 = 2\*count2;  
plot\_3Djpd(rand\_num,100,count1,count2);  
  
% Establish the given covariance matrices and mean for part two  
mu2 = [6 6];  
A = [1 0; 0 1];  
B = [5 0; 0 2];  
C = [1 0.5; 0.5 1];  
D = [5 0.5; 0.5 2];  
  
% Find 2D normally distributed random vector q from given information  
q = cov\_randgen(A,mu2);  
  
% Plot the 3D Probability Density Function of the random vector q  
count1 = (2\*count1) + 1;  
count2 = 2\*count2;  
plot\_3Djpd(q,100,count1,count2);  
  
% Repeat for the remaining covariance matrices  
q = cov\_randgen(B,mu2);  
count1 = (2\*count1) + 1;  
count2 = 2\*count2;  
plot\_3Djpd(q,100,count1,count2);  
  
q = cov\_randgen(C,mu2);  
count1 = (2\*count1) + 1;  
count2 = 2\*count2;  
plot\_3Djpd(q,100,count1,count2);  
  
q = cov\_randgen(D,mu2);  
count1 = (2\*count1) + 1;  
count2 = 2\*count2;  
plot\_3Djpd(q,100,count1,count2);  
  
end  
  
% This function generates a 2D array of random numbers in the range of  
% [0,1] based on the input size and the type of distribution desired  
%  
% Input(s): size, random number generator type  
% Output(s): array of random numbers  
function x = generate\_2Drand(type,size)  
  
if(strcmp(type,'Uniform') || strcmp(type,'uniform'))  
  
 x = rand(size,2);  
  
elseif(strcmp(type,'Normal') || strcmp(type,'normal'))  
  
 x = randn(size,2);  
  
end  
  
end  
  
% This function first finds the Joint Probability Distribution of  
% argument y then plots it on a 3D graph  
%  
% Input(s): 2D array a, distribution type  
% Output(s): surf plot  
function x = plot\_3Djpd(a,nbins,count1,count2)  
  
% Separate x and y from input data set a  
x0 = a(1:length(a),1);  
y0 = a(1:length(a),2);  
  
% Find the histogram of each axis and isolate the values  
figure(count1);  
hx0 = histogram(x0,nbins);  
hx0.Normalization = 'probability';  
hold on;  
hy0 = histogram(y0,nbins);  
hy0.Normalization = 'probability';  
title('Probability Mass Functions w.r.t (X,0,Z) and (0,Y,Z)');  
legend('(X,0,Z)','(0,Y,Z)');  
  
xpdf = hx0.Values;  
ypdf = hy0.Values;  
  
% Since the two dimensions are independent we can find the joint  
% probability distribution function by multiplying the PMF of x and y  
jpd = ((xpdf.')\*ypdf);  
  
% Plot the jpd in three dimensions with respect to the PMF of x and y  
figure(count2);clf;  
surf(jpd);  
title('Joint Probability Distribution of 2D Array')  
  
% Demonstrate that the JPD is true and sums to one for all values of x and  
% y  
test = sum(jpd);  
test = sum(test);  
str = sprintf('The sum of all the z magnitudes in the Joint Probability Distribution is %d', test);  
disp(str);  
  
end  
  
% This function uses the mvnrnd() function present in MATLAB to generate  
% two vectors of random numbers with a uniform distribution based on an  
% input covariance matrix and input means for outputs x and y  
%  
% Input(s): M covariance matrix between x and y, m means of x and y  
% Output(s): x and y normally distributed random variables  
function x = cov\_randgen(M,m)  
  
% Set x and y to have a length of 1000 samples each  
cases = 1e3;  
  
% Call the multivariate random number generation function  
x = mvnrnd(m,M,cases);  
  
end

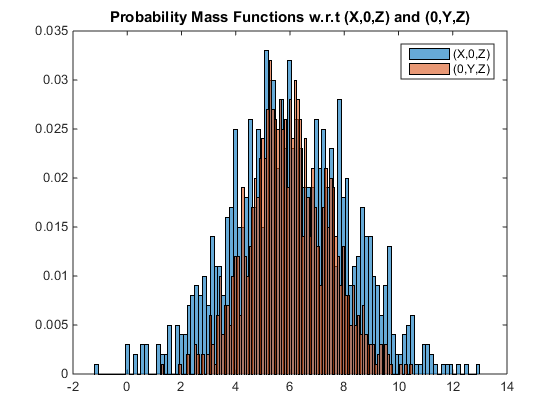
The sum of all the z magnitudes in the Joint Probability Distribution is 1.000000e+00  
The sum of all the z magnitudes in the Joint Probability Distribution is 1.000000e+00  
The sum of all the z magnitudes in the Joint Probability Distribution is 1.000000e+00  
The sum of all the z magnitudes in the Joint Probability Distribution is 1.000000e+00  
The sum of all the z magnitudes in the Joint Probability Distribution is 1.000000e+00

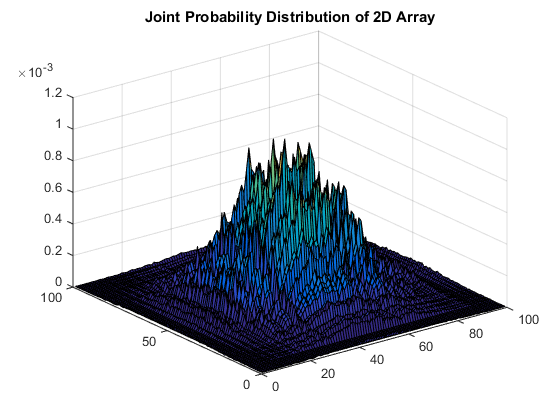


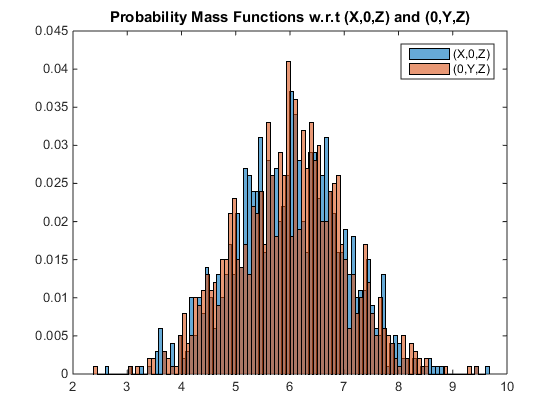


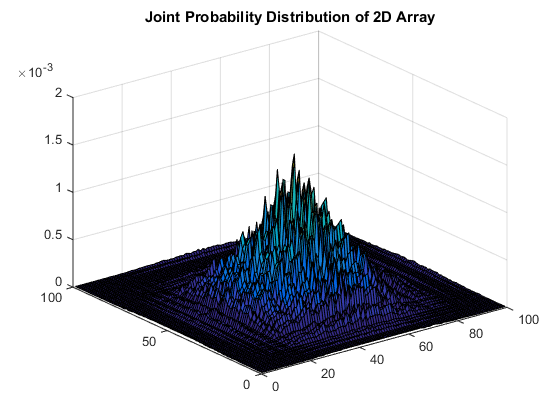


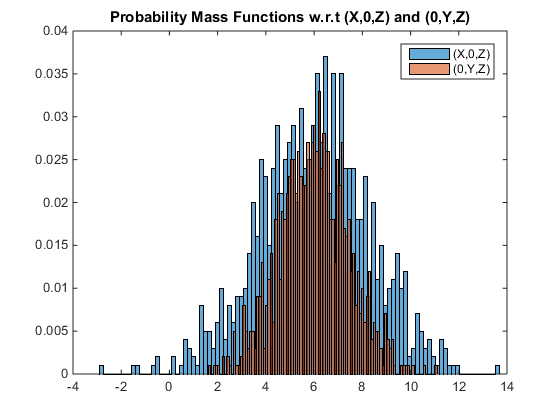


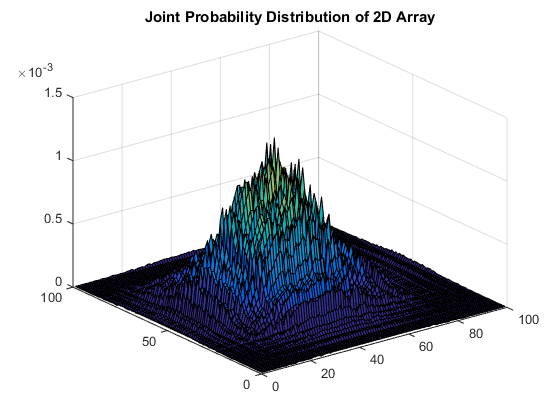






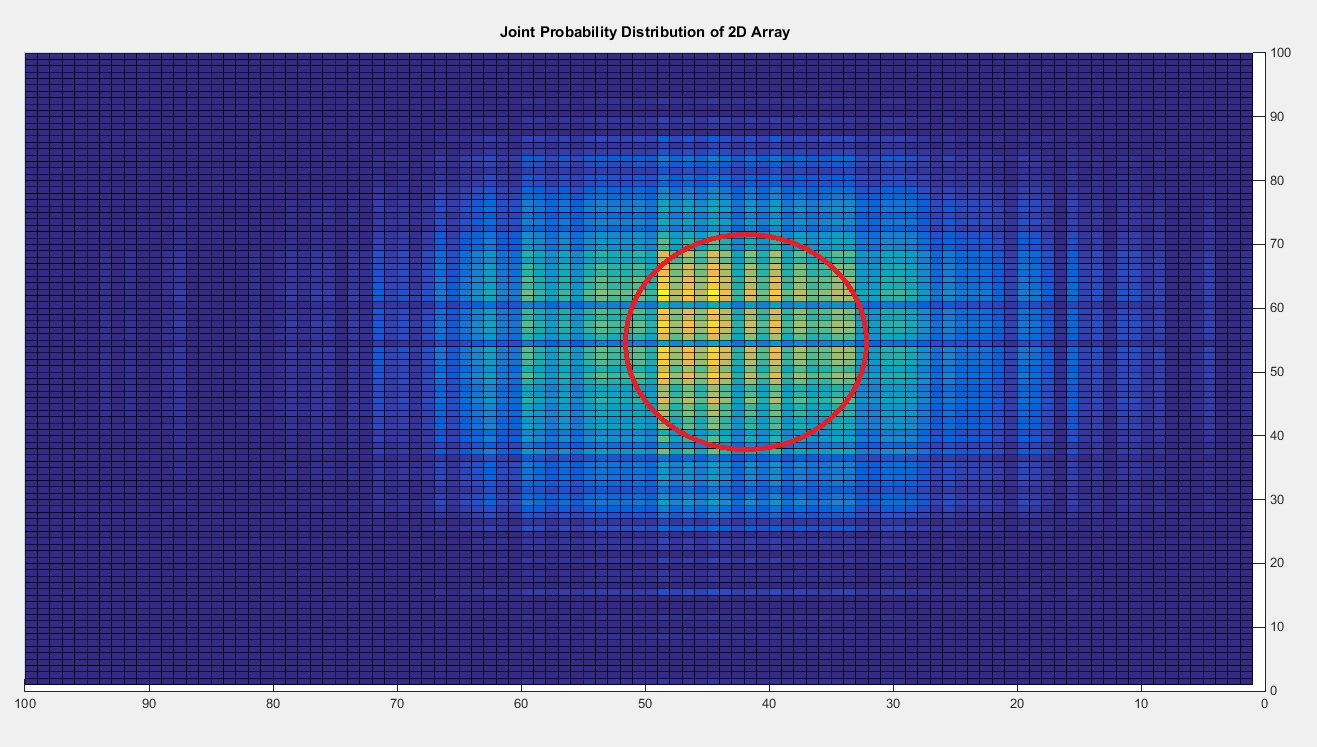


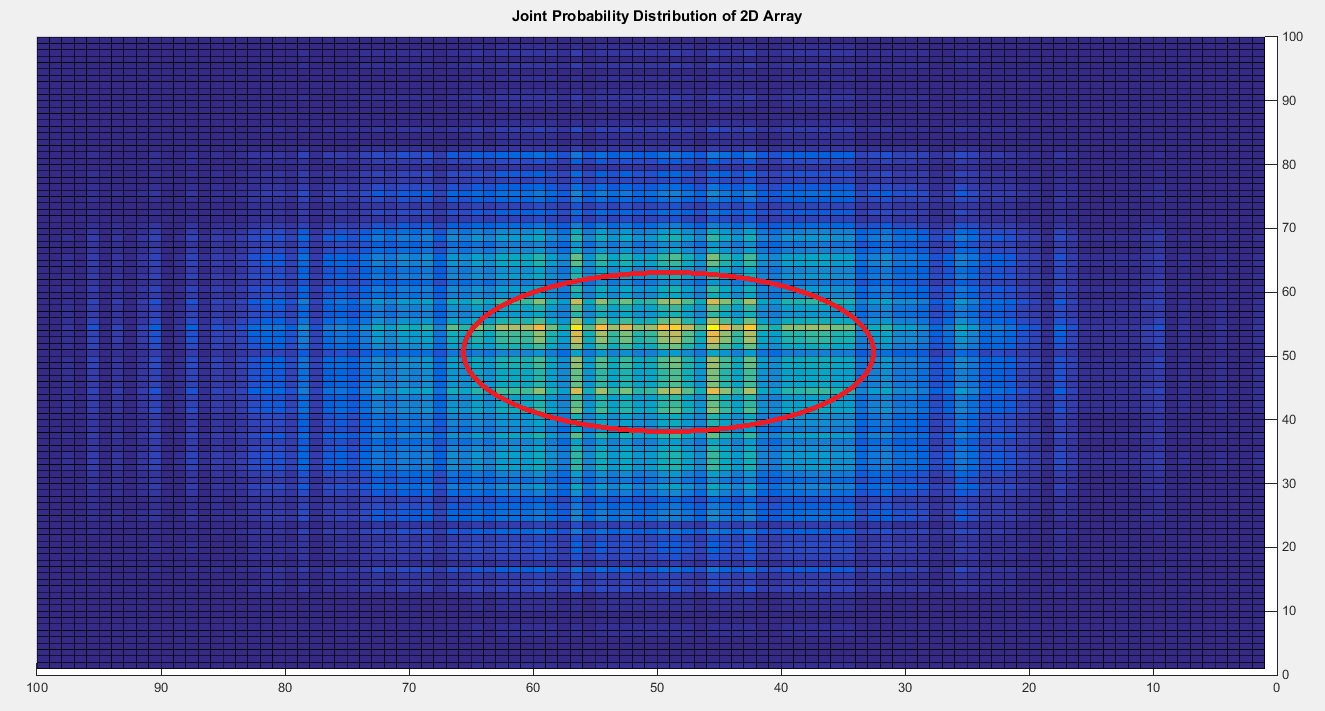


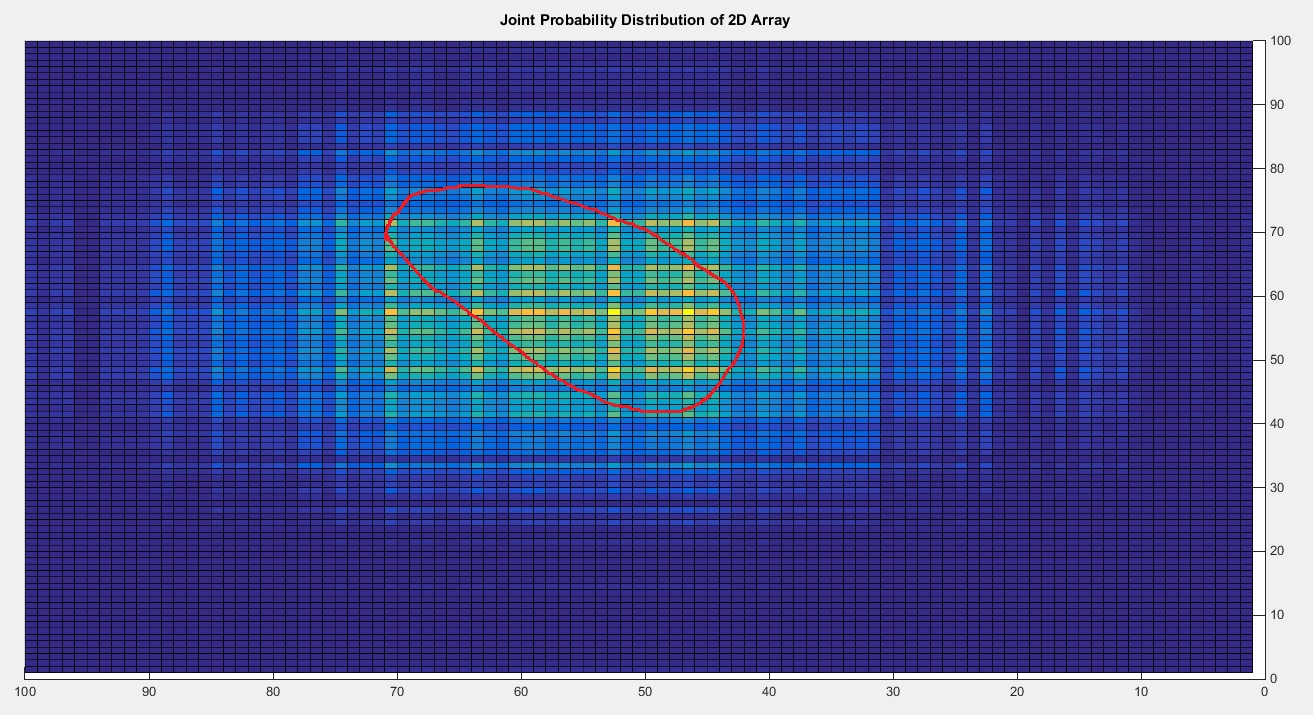


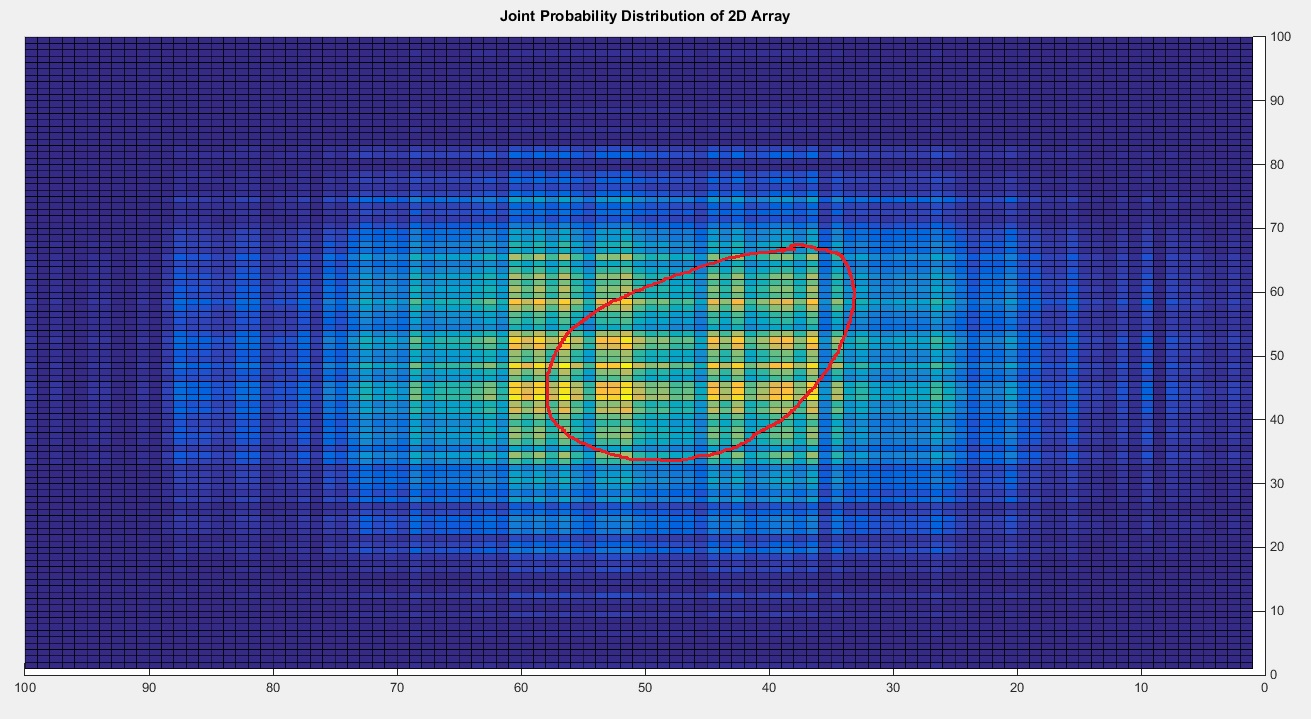
[*Published with MATLAB® R2014b*](http://www.mathworks.com/products/matlab)

The following cross sections of the Gaussian distributions have had circles drawn on them to facilitate understanding of the effect of the covariance matrix on the generated random variables.









# Conclusions

From the joint probability distributions, it is clear that generated random variables, while imperfect, did behave as we expected them to. The identity matrix produced a support region that was fairly circular in appearance whereas multiplying the main diagonal with constants stretched and compressed the support region and varying the correlation values in the reverse diagonal produced rotation of the support region. Additionally, all of the joint probability distributions summed to one for the double sum in the x and y direction which tells us that these are valid distributions to use. One thing I was not able to figure out is why the individual PDFs for the and random vectors had means of six as they were intended to whilst the joint probability distribution seems to be centered at fifty. Perhaps this has something to do with the length of the vectors and their dot product but debugging along those lines has been inconclusive. Overall, the Gaussian joint probability distributions behave as they are supposed to which is the focus of this assignment.