# NUEN 647
# Uncertainty Quantification for Nuclear Engineering
# Homework 2

Due on Wednesday, October 19, 2016

*Dr. McClarren*

**Paul Mendoza**

# Contents

# Problem 1

Consider a covariance function between points in 2-D space:

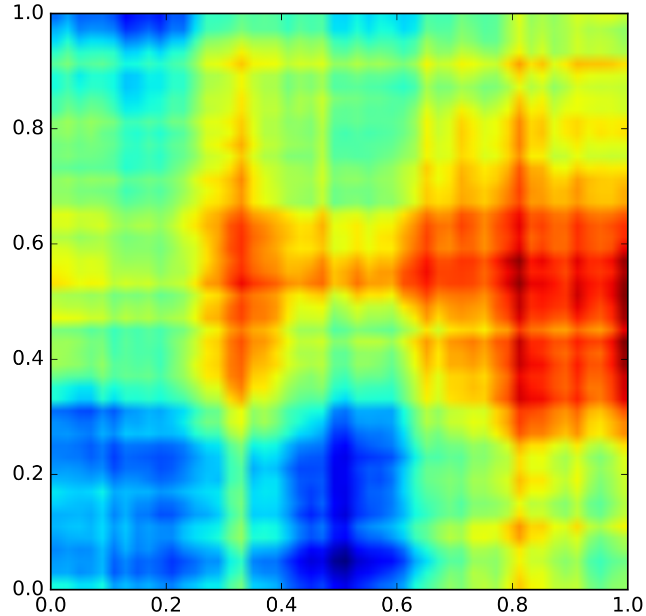
$$k(x_1, y_1, x_2, y_2) = exp[-|x_1 - x_2| - |y_1 - y_2|]$$

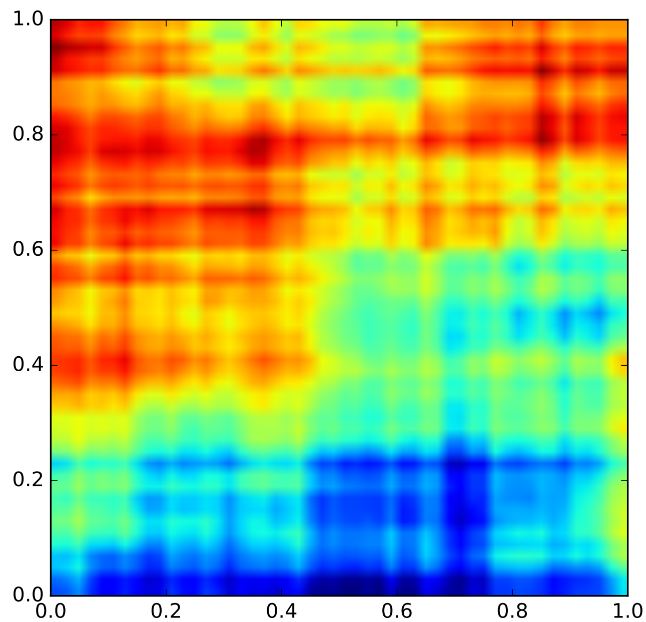
Generate 4 realizations of a Gaussian random process with zero mean, $\mu(x, y) = 0$, and this covariance function defined on the unit square, $x, y \in [0, 1]$. For the realizations, evaluate the process at 50 equally space points in each direction. Plot the realizations.

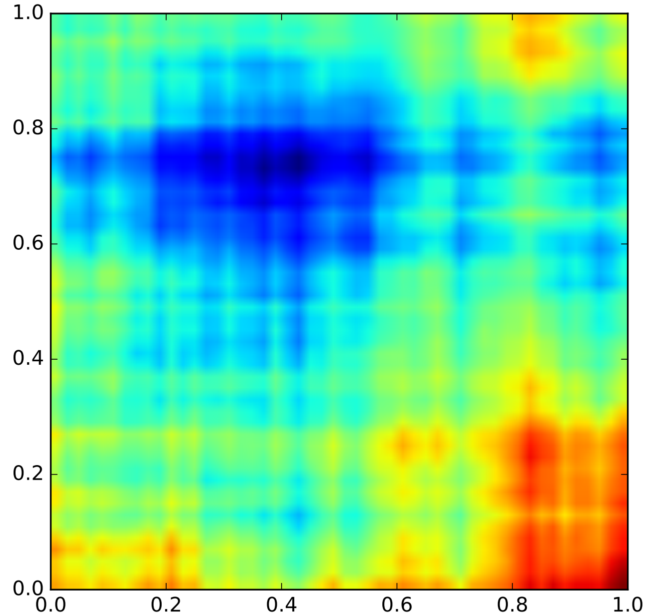
I received lots of help on this problem from other members in the class. Problem took around 400 seconds to run.

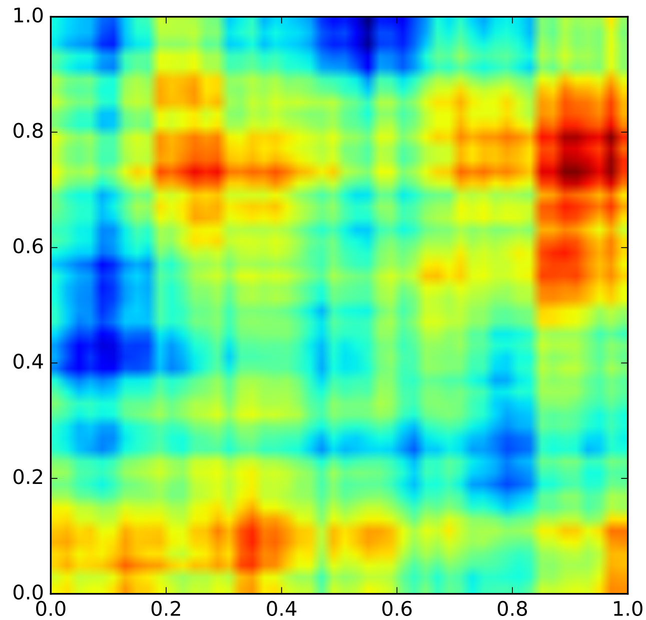
Listing 1: Script for Problem

```
#!/usr/bin/env python3

#################################################################
##################### Import packages ##########################
#################################################################

import numpy as np
import matplotlib.pyplot as plt
import time
start_time = time.time()
from scipy.stats import multivariate_normal

import Functions as fun

#################################################################
###################### Calculations ############################
#################################################################

u=np.zeros(2500)

k=np.zeros((2500,2500))

for i in range(0,50):
    print(i)
    for j in range(0,50):
        for l in range(0,50):
            for m in range(0,50):
                k[i*50+l,j*50+m] = np.exp(-np.abs(i/50.0-j/50.0)-np.abs(l/50.0-m/50.0))

for m in range(1,5):
    U=multivariate_normal.rvs(u,k)
    Z = [[U[50*i+j] for j in range(0,50)] for i in range(0,50)]
    plt.imshow(Z, extent=(0,1,0,1))
    plt.savefig("P1realization"+str(m)+".pdf")

##################### Time To execute #################

print("--- %s seconds ---" % (time.time() - start_time))
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

# Problem 2

Assume you have 100 samples of a pair of random variables $(X_1, X_2)$ that have a positive correlation, call this set of pairs, $\mathbf{A_1}$. You then draw another 100 samples and call this set $\mathbf{A_2}$. The Pearson correlation between $(X_1, X_2)$ in $\mathbf{A_1}$ is positive and hte Pearson correlation between $(X_1, X_2)$ in $\mathbf{A_2}$ is negative. What can you say about the Pearson correlation for all 200 samples?

A normalized measure of the relation between two random variables, is the Pearson correlation coefficient, $\rho$. Oftentimes, this is simply called the correlation coefficient or correlation.

$$\rho(X_1, X_2) = \frac{E[X_1 X_2] - E[X_1]E[X_2]}{\sigma_{X1}\sigma_{X2}}$$

The expectation value for a series of realizations is defined:

$$E[g(x)] \approx \frac{1}{N}\sum_{i=1}^{N} g(x_i)$$

For the first 100 values:

$$\rho_1 = \frac{\frac{1}{100}\sum_{i=1}^{100} X_{1,i}X_{2,i} - \frac{1}{10000}\sum_{i=1}^{100} X_{1,i}\sum_{i=1}^{100} X_{2,i}}{\sigma_{X1,A1}\sigma_{X2,A1}}$$

$$100\sigma_{X1,A1}\sigma_{X2,A1}\rho_1 = \sum_{i=1}^{100} X_{1,i}X_{2,i} - \frac{1}{100}\sum_{i=1}^{100} X_{1,i}\sum_{i=1}^{100} X_{2,i}$$

$$100\sigma_{X1,A1}\sigma_{X2,A1}\rho_1 + \frac{1}{100}\sum_{i=1}^{100} X_{1,i}\sum_{i=1}^{100} X_{2,i} = \sum_{i=1}^{100} X_{1,i}X_{2,i}$$

Similarly for the second 100 values:

$$\sum_{i=101}^{200} X_{1,i}X_{2,i} = 100\sigma_{X1,A2}\sigma_{X2,A2}\rho_2 + \frac{1}{100}\sum_{i=101}^{200} X_{1,i}\sum_{i=101}^{200} X_{2,i}$$

The Pearson coefficient for all 200 values:

$$\rho_3 = \frac{\frac{1}{200}\sum_{i=1}^{200} X_{1,i}X_{2,i} - \frac{1}{40000}\sum_{i=1}^{200} X_{1,i}\sum_{i=1}^{200} X_{2,i}}{\sigma_{X1,A3}\sigma_{X2,A3}}$$

$$200\sigma_{X1,A3}\sigma_{X2,A3}\rho_3 = \sum_{i=1}^{200} X_{1,i}X_{2,i} - \frac{1}{200}\sum_{i=1}^{200} X_{1,i}\sum_{i=1}^{200} X_{2,i}$$

$$200\sigma_{X1,A3}\sigma_{X2,A3}\rho_3 + \frac{1}{200}\sum_{i=1}^{200} X_{1,i}\sum_{i=1}^{200} X_{2,i} = \sum_{i=1}^{200} X_{1,i}X_{2,i}$$

If we plug in the Pearson for the first 100 and second 100 for the right side of the equation,

Problem 2 continued on next page. . . 

$$200\sigma_{X1,A3}\sigma_{X2,A3}\rho_3 + \frac{1}{200}\sum_{i=1}^{200} X_{1,i}\sum_{i=1}^{200} X_{2,i}$$
$$=$$
$$100(\sigma_{X1A1}\sigma_{X2A1}\rho_1 + \sigma_{X1A2}\sigma_{X2A2}\rho_2) + \frac{1}{100}\left(\sum_{i=1}^{100} x_{1,i}\sum_{i=1}^{100} x_{2,i} + \sum_{i=101}^{200} x_{1,i}\sum_{i=101}^{200} x_{2,i}\right)$$

Grouping and setting:

$$\textstyle\sum_{i=1}^{100} x_{1,i} = X_{1,1}$$
$$\textstyle\sum_{i=1}^{100} x_{2,i} = X_{2,1}$$
$$\textstyle\sum_{i=101}^{200} x_{1,i} = X_{1,2}$$
$$\textstyle\sum_{i=101}^{200} x_{2,i} = X_{2,2}$$
$$\textstyle\sum_{i=1}^{200} x_{1,i} = X_{1,3}$$
$$\textstyle\sum_{i=1}^{200} x_{2,i} = X_{2,3}$$

$$200\sigma_{X1,A3}\sigma_{X2,A3}\rho_3 - 100(\sigma_{X1A1}\sigma_{X2A1}\rho_1 + \sigma_{X1A2}\sigma_{X2A2}\rho_2) = \frac{1}{100}\left(X_{1,1}X_{2,1} + X_{1,2}X_{2,2}\right) - \frac{1}{200}X_{1,3}X_{2,3}$$

Setting:

$$\sigma_{X1A1} = \tfrac{1}{100}\textstyle\sum_{i=1}^{100}(x_{1,i}^2 - \mu_{X_{11}}^2) = \tfrac{1}{100}\sigma'_{X1A1}$$
$$\sigma_{X2A1} = \tfrac{1}{100}\textstyle\sum_{i=1}^{100}(x_{2,i}^2 - \mu_{X_{21}}^2) = \tfrac{1}{100}\sigma'_{X2A1}$$
$$\sigma_{X1A2} = \tfrac{1}{100}\textstyle\sum_{i=101}^{200}(x_{1,i}^2 - \mu_{X_{12}}^2) = \tfrac{1}{100}\sigma'_{X1A2}$$
$$\sigma_{X2A2} = \tfrac{1}{100}\textstyle\sum_{i=101}^{200}(x_{2,i}^2 - \mu_{X_{22}}^2) = \tfrac{1}{100}\sigma'_{X2A2}$$
$$\sigma_{X1A3} = \tfrac{1}{200}\textstyle\sum_{i=1}^{200}(x_{1,i}^2 - \mu_{X_{13}}^2) = \tfrac{1}{200}\sigma'_{X1A3}$$
$$\sigma_{X2A3} = \tfrac{1}{200}\textstyle\sum_{i=1}^{200}(x_{2,i}^2 - \mu_{X_{23}}^2) = \tfrac{1}{200}\sigma'_{X2A3}$$

Where $A3$ and $\rho_3$ are for the series added to 200. Plugging these in, and multiplying both sides of the equation by 200.

$$\sigma'_{X1,A3}\sigma'_{X2,A3}\rho_3 - 2(\sigma'_{X1A1}\sigma'_{X2A1}\rho_1 + \sigma'_{X1A2}\sigma'_{X2A2}\rho_2) = 2\left(X_{1,1}X_{2,1} + X_{1,2}X_{2,2}\right) - X_{1,3}X_{2,3}$$

Note: $X_{1,3} = X_{1,1} + X_{2,1}$ and $X_{2,3} = X_{2,1} + X_{2,2}$ and that the right side of the equation simplifies to: $(X_{1,1} - X_{1,2})(X_{2,1} - X_{2,2})$. Then $\rho_3$ is:

$$\rho_3 = \frac{(X_{1,1} - X_{1,2})(X_{2,1} - X_{2,2}) + 2(\sigma'_{X1A1}\sigma'_{X2A1}\rho_1 + \sigma'_{X1A2}\sigma'_{X2A2}\rho_2)}{\sigma'_{X1,A3}\sigma'_{X2,A3}}$$

Assuming that:

$$\frac{(X_{1,1} - X_{1,2})(X_{2,1} - X_{2,2})}{\sigma'_{X1,A3}\sigma'_{X2,A3}} \approx 0$$

and

$$\begin{aligned} \sigma'_{X1,A3}\sigma'_{X2,A3} &\approx 4\sigma'_{X1A1}\sigma'_{X2A1} \\ or &\approx 4\sigma'_{X1A2}\sigma'_{X2A2} \end{aligned}$$

The above would simplify to:

$$\rho_3 \approx \frac{\rho_1 + \rho_2}{2}$$

Meaning, $\rho_3$ will usually be inside the interval $\rho_2 < \rho_3 < \rho_1$, I was curious, and wrote a script, to check to see if it would ever be outside. There could be an error with my script, but I found with the below script that a small percentage (less than 1% of the time around 0.2-0.4%), it would be outside the above interval. I also made a histogram plot...because I like wasting time.

Listing 2: Script for Problem

```
#!/usr/bin/env python3

###############################################################
#################### Import packages ##########################
###############################################################

import numpy as np
import time
start_time = time.time()
import Functions as Fun

###############################################################
#################### Calculations #############################
###############################################################

Error=[];Ntimes=1000;Nsamples=100;CountOut=0

for i in range(0,Ntimes):

    Positive=True
    Negative=True
    while(Positive or Negative):

        X1=np.random.uniform(-1,1,Nsamples)
        X2=np.random.uniform(-1,1,Nsamples)
        rho=Fun.CalculateRho(X1,X2)

        if rho>0:
            rho1=rho;X11=X1;X21=X2;
            Positive=False
        if rho<0:
            rho2=rho;X12=X1;X22=X2;
```

```
            Negative=False


    rho_Guess=(rho1+rho2)/2

    X13=np.append(X11,X12)
    X23=np.append(X21,X22)
    rho=Fun.CalculateRho(X13,X23)

    if(rho>rho1 or rho<rho2):
        CountOut=CountOut+1
    Error.append((abs(rho_Guess-rho)/rho)*100)

Fun.PlotHistSave(Error,Ntimes)

print("Percent outside rho1 and rho2: "+str(100*CountOut/Ntimes)+"%")

##################### Time To execute ################

print("--- %s seconds ---" % (time.time() - start_time))
```

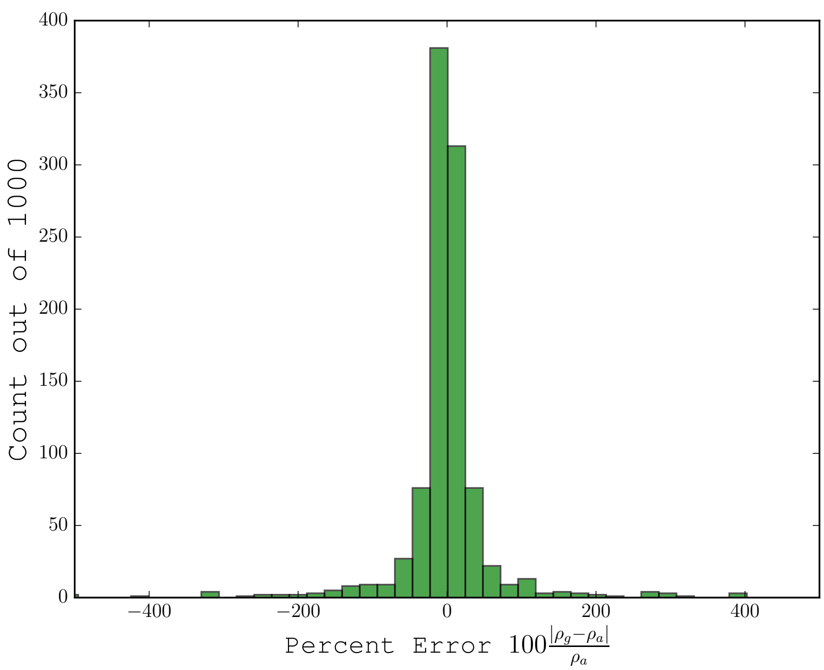



Figure 1: Histogram plot showing error $\rho_g$ is the approximated guess at $\rho_3$ and $\rho_a$ is the actual calculated $\rho_3$.

# Problem 3

For the following data, compute by hand or via code you write the Pearson and Spearman correlations and Kendall's tau.

| $X_1$ | $X_2$ |
|---|---|
| 55.01 | 82.94 |
| 54.87 | 55.02 |
| 57.17 | 85.18 |
| 36.01 | -84.27 |
| 35.88 | -106.30 |
| 36.33 | -119.65 |
| 43.49 | -112.03 |
| 41.44 | -71.69 |
| 54.43 | -3.50 |
| 36.47 | 140.57 |

**Pearson Correlation**

$$\rho(X,Y) = \frac{E[XY] - E[X]E[Y]}{\sigma_X \sigma_Y}$$

Where:

$$E[g(x)] = \int_{-\infty}^{\infty} dx\ g(x) f(x) \approx \frac{1}{N}\sum_{i=1}^{N} g(x_i)$$

and

$$\sigma_X = Var(x) = \int_{-\infty}^{\infty} (x-\mu)^2 f(x) dx \approx \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \approx \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 \equiv s^2$$

and

$$\mu_X \approx \frac{1}{N}\sum_{i=1}^{N} x_i \equiv \bar{x}$$

**Spearman Rank Correlation**

$$\rho_S(X,Y) = \frac{\Sigma_{i=1}^{N}(rank(x_i) - \bar{r}_X)(rank(y_i) - \bar{r}_Y)}{\sqrt{\Sigma_{i=1}^{N}(rank(x_i) - \bar{r}_X)^2}\sqrt{\Sigma_{i=1}^{N}(rank(y_i) - \bar{r}_Y)^2}}$$

Where:

$rank(x_i)$= Rank of $x_i$ in sample population

and

$$\bar{r}_X = \frac{1}{N}\Sigma_{i=1}^{N} rank(x_i)$$

**Kendall's Tau**

**TAU!!!!!!** (Powering up)

$$\tau = \frac{\text{(\# of concordant pairs) - (\# of discordant pairs)}}{\frac{1}{2}N(N-1)}$$

Where concordance is

$$x_i > x_j \text{ and } y_i > y_j \text{ or if } x_i < x_j \text{ and } y_i < y_j$$

for all pairs of samples ($\frac{1}{2}N(N-1)$ of them) and discordance is

$$x_i > x_j \text{and } y_i < y_j \text{ or if } x_i < x_j \text{and } y_i > y_j$$

Listing 3: Script for Problem

```
#!/usr/bin/env python3

###############################################################
#################### Import packages #########################
###############################################################
import numpy as np
import time
start_time = time.time()
import Functions as Fun
###############################################################
#################### Calculations ############################
###############################################################

X1=np.array([55.01,54.87,57.17,36.01,35.88,36.33,
             43.49,41.44,54.43,36.47])
X2=np.array([82.94,55.02,85.18,-84.27,-106.30,-119.65,
             -112.03,-71.69,-3.50,140.57])

rho,rhoNM1=Fun.CalculatePearson(X1,X2)

#Getting rank of each element, starting with 1
X1R=Fun.Rank(X1)
X2R=Fun.Rank(X2)

rhoS=Fun.CalculateSpearman(X1,X2,X1R,X2R)
tau=Fun.CalculateTau(X1,X2)

print("Pearson Var Div by N: "+str(round(rho,4)))
print("Pearson Var Div by N-1: "+str(round(rhoNM1,4)))
print("Spearman: "+str(round(rhoS,4)))
print("Kendall: "+str(round(tau,4)))
##################### Time To execute ################
print("--- %s seconds ---" % (time.time() - start_time))
```

Code output:

Pearson Var Div by N: $\boxed{0.5429}$
Pearson Var Div by N-1: $\boxed{0.4886}$
Spearman: $\boxed{0.5879}$
Kendall: $\boxed{0.5111}$

# Problem 4

Demonstrate the tail dependence of a bivariate normal random variable is 0.

The bivariate Gaussian copula is defined as:

$$C_N(u,v) = \Phi_\rho(\Phi^{-1}(u), \Phi^{-1}(v))$$

Where:

$$\Phi^{-1}(q) = \mu + \sigma\sqrt{2}erf^{-1}(2q-1)$$

Evaluated at $q=0$ and $q=1$:

$$\Phi^{-1}(0) = -\infty \qquad \Phi^{-1}(1) = \infty$$

Also where:

$$\Phi_\rho(x,y) = \int_{-\infty}^{x} dx' \int_{-\infty}^{y} dy' \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} exp\left[-\frac{z}{2(1-\rho^2)}\right]$$

with

$$z = \frac{(x'-\mu_x)^2}{\sigma_x^2} - \frac{2\rho(x'-\mu_x)(y'-\mu_y)}{\sigma_x\sigma_y} + \frac{(y'-\mu_y)^2}{\sigma_y^2}$$

and

$$\rho = \frac{E[XY]-E[X]E[Y]}{\sigma_x\sigma_y}$$

**Note:**

$$\Phi_\rho(-\infty,-\infty) = \int_{-\infty}^{-\infty} dx' \int_{-\infty}^{-\infty} dy' \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} exp\left[-\frac{z}{2(1-\rho^2)}\right] = 0$$

Because integrating over zero domain is 0.

$$\Phi_\rho(\infty,\infty) = \int_{-\infty}^{\infty} dx' \int_{-\infty}^{\infty} dy' \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} exp\left[-\frac{z}{2(1-\rho^2)}\right] = 1$$

Because integrating over the entire domain of a PDF is 1.

**Lower Tail Dependance:**

$$\begin{aligned}\lambda_l &= \lim_{q\to 0} \frac{C(q,q)}{q} \\ &= \lim_{q\to 0} \frac{\Phi_\rho(\Phi^{-1}(q), \Phi^{-1}(q))}{q}\end{aligned}$$

Applying L'Hôspital (Pronouced Hospital - like the place you go when you get sick - not really).

$$\begin{aligned}\lambda_l &= \lim_{q\to 0} \frac{\frac{d}{dq}\Phi_\rho(\Phi^{-1}(q), \Phi^{-1}(q))}{1} \\ &= \lim_{q\to 0} \frac{d}{dq}\Phi_\rho(\Phi^{-1}(q), \Phi^{-1}(q))\end{aligned}$$

Evaluating at $q = 0$

$$\begin{aligned}
\lambda_l &= \frac{d}{dq}\Phi_\rho(\Phi^{-1}(0), \Phi^{-1}(0)) \\
&= \frac{d}{dq}\Phi_\rho(-\infty, -\infty) \\
&= \frac{d}{dq}0 \\
&= 0
\end{aligned}$$

The problem with this is, I do not think I can just plug in $q = 0$ inside the whole mess, but rather like this:

$$\lambda_l = \left|\frac{d}{dq}\Phi_\rho(\Phi^{-1}(q), \Phi^{-1}(q))\right|_{q=0}$$

Here, I would want to say that the integral cancels with $\frac{d}{dq}$ (not sure if thats correct - because there are two integrals), to get

$$\begin{aligned}
\lambda_l &= \left|\phi(\Phi^{-1}(q), \Phi^{-1}(q))\right|_{q=0} \\
&= \phi(\Phi^{-1}(0), \Phi^{-1}(0)) \\
&= \phi(-\infty, -\infty) \\
&= 0
\end{aligned}$$

Where $\phi$ is the PDF of a bivariate normal ($\Phi$ without the integrals). Noting $\phi(-\infty, -\infty)$ ends up with a $e^{-\infty}$ term.

but if we apply the limit before L'Hôspital, then we get:

$$\begin{aligned}
\lambda_l &= \lim_{q\to 0}\frac{C(q,q)}{q} \\
&= \lim_{q\to 0}\frac{\Phi_\rho(\Phi^{-1}(q), \Phi^{-1}(q))}{q} \\
&= \frac{\Phi_\rho(\Phi^{-1}(0), \Phi^{-1}(0))}{0} \\
&= \frac{\Phi_\rho(-\infty, -\infty)}{0} \\
&= \frac{0}{0}
\end{aligned}$$

I am not sure if thats healthy.

**Upper Tail Dependance:**

$$\begin{aligned}
\lambda_u &= \lim_{q\to 1} \frac{1-2q+C(q,q)}{1-q} \\
&= \lim_{q\to 1} \frac{1-2q+\Phi_\rho(\Phi^{-1}(q),\Phi^{-1}(q))}{1-q}
\end{aligned}$$

Applying the limit:

$$\begin{aligned}
\lambda_u &= \frac{1-2+\Phi_\rho(\Phi^{-1}(1),\Phi^{-1}(1))}{0} \\
&= \frac{1-2+\Phi_\rho(\infty,\infty)}{0} \\
&= \frac{1-2+1}{0} \\
&= \frac{0}{0}
\end{aligned}$$

Again, I do not think this is healthy, but I don't know what else to do. Could look at similar things as above, but I think they wouldn't even give 0...I could try and evaluate the intervals and then differentiate but that sounds like a mess.

# Problem 5

Another Archimedean copula is the Joe copula with generator

$$\varphi_J(t) = -log(1-(1-t)^\theta),$$

and

$$\varphi_J^{-1} = 1-(1-exp(-t))^{1/\theta}.$$

(a) Compute the bivariate copula for this generator
The Archimedean copula for $\varphi$ is

$$C_\varphi(u,v) = \hat{\varphi}^{-1}(\varphi(u)+\varphi(v))$$

(b) Derive the upper and lower tail dependence for this copula

$$\lambda_l = \lim_{q\to 0}\frac{C(q,q)}{q}$$

$$\lambda_u = \lim_{q\to 1}\frac{1-2q+C(q,q)}{1-q}$$

(c) Compute the value of Kendall's tau for this copula

$$\tau(U,V) = 1+4\int_0^1 \frac{\varphi(t)}{\varphi'(t)}dt$$

(d) Generate 1000 samples from the copula with standard normal margins and a value of Kendall's tau of 0.6.

**(a)**

$$\begin{aligned}
C_\varphi(u,v) &= \hat{\varphi}^{-1}(\varphi(u)+\varphi(v))\\
&= \hat{\varphi}^{-1}(-log(1-(1-u)^\theta) - log(1-(1-v)^\theta))\\
&= \hat{\varphi}^{-1}(-log([1-(1-u)^\theta][1-(1-v)^\theta]))\\
&= 1-(1-exp(-\left(-log([1-(1-u)^\theta][1-(1-v)^\theta]))\right))^{1/\theta}\\
&= 1-(1-[1-(1-u)^\theta][1-(1-v)^\theta])^{1/\theta}\\
&= 1-\left[(1-v)^\theta+(1-u)^\theta-(1-u)^\theta(1-v)^\theta\right]^{1/\theta}
\end{aligned}$$

**(b) Lower**

$$\begin{aligned}
\lambda_l &= \lim_{q\to 0}\frac{C(q,q)}{q}\\
&= \lim_{q\to 0}\frac{1-\left[(1-q)^\theta+(1-q)^\theta-(1-q)^\theta(1-q)^\theta\right]^{1/\theta}}{q}\\
&= \lim_{q\to 0}\frac{1-\left[2(1-q)^\theta-(1-q)^{2\theta}\right]^{1/\theta}}{q}\\
&= \lim_{q\to 0}\frac{1-\left[(1-q)^\theta(2-(1-q)^\theta)\right]^{1/\theta}}{q}\\
&= \lim_{q\to 0}\frac{1-(1-q)\left[(2-(1-q)^\theta)\right]^{1/\theta}}{q}
\end{aligned}$$

Problem 5 continued on next page... 

Applying L'Hôspital (did it in my head - not really)

$$\begin{aligned}\lambda_l &= \lim_{q\to 0} -2(2-(1-q))^{\frac{1}{\theta}-1}((1-q)^\theta - 1)\\ &= -2(2-(1-0))^{\frac{1}{\theta}-1}((1-0)^\theta - 1)\\ &= 0\end{aligned}$$

**Upper**

$$\begin{aligned}\lambda_u &= \lim_{q\to 1} \frac{1-2q+C(q,q)}{1-q}\\ &= \lim_{q\to 1} \frac{1-2q+1-(1-q)\left[(2-(1-q)^\theta)\right]^{1/\theta}}{1-q}\\ &= \lim_{q\to 1} \frac{(1-q)\left(2-[(2-(1-q)^\theta)]^{1/\theta}\right)}{1-q}\\ &= \lim_{q\to 1} 2-\left[(2-(1-q)^\theta)\right]^{1/\theta}\\ &= 2-2^{1/\theta}\end{aligned}$$

**(c)**

$$\varphi'(t) = -\frac{\theta(1-t)^{\theta-1}}{1-(1-t)^\theta}$$

$$\begin{aligned}\tau(U,V) &= 1+4\int_0^1 \frac{\varphi(t)}{\varphi'(t)}dt\\ &= 1+4\int_0^1 \frac{-log(1-(1-t)^\theta)}{-\frac{\theta(1-t)^{\theta-1}}{1-(1-t)^\theta}}dt\\ &= 1+4\int_0^1 \frac{-log(1-(1-t)^\theta)}{-\frac{\theta(1-t)^{\theta-1}}{1-(1-t)^\theta}}dt\end{aligned}$$

I do not want to integrate, and will assume the answer is correct on this link.

$$\tau(U,V) = 1-4\sum_{k=1}^{\infty}\frac{1}{(K(\theta K+2)(\theta(K-1)+1))}$$

**(d)** For a Kendall's tau of 0.6, $\theta = 3.826659$. According to some random notes I have. In order to sample from a bivariate copula:

1. Produce $\xi_1$, $\xi_2$ where $\xi_i \sim U(0,1)$
2. Set $W \equiv C^{-1}(\xi_2|\xi_1)$

   Where:

$$\begin{aligned} C(v|u) &= \frac{d}{du}(C(u,v)) \\ &= \frac{d}{du}(1 - \left[(1-v)^\theta + (1-u)^\theta - (1-u)^\theta(1-v)^\theta\right]^{1/\theta}) \end{aligned}$$

   Wolfram gives an answer, but I want to say first, this is annoying. Okay, if $U = (1-u)^\theta$ and $V = (1-v)^\theta$ and $U^* = (1-u)$, then a reasonable looking answer is:

$$C(v|u) = \frac{U}{U^*}[V-1][U+V-VU]^{\frac{1}{\theta}-1}$$

   The next step requires us setting $C(v|u) = \xi$ and solving for $v$ and calling that $C_J^{-1}(\xi|u)$. I cannot seem to solve for V, so I'll come up with a janky way to determine $C_J^{-1}(\xi|u)$. Solving for a $V$, and remembering that $V = (1-v)^\theta = (1 - C_J^{-1}(\xi|u))^\theta$.

$$V = \frac{U^*}{U}\xi[U+V-VU]^{1-\frac{1}{\theta}} + 1$$

$$C_J^{-1}(\xi|u) = 1 - \left(\frac{U}{U^*}\xi[U+V-VU]^{1-\frac{1}{\theta}} + 1\right)^{\frac{1}{\theta}}$$

   Noting that we will have to iterate for a solution... this better converge, or I will destroy something.

3. $x = F_X^{-1}(\xi_1) \quad y = F_Y^{-1}(w)$ Where:

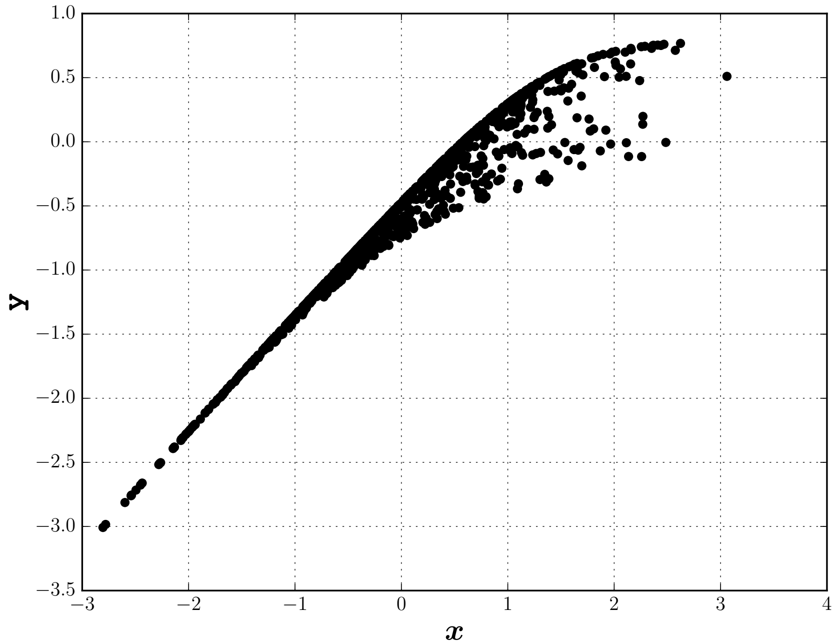
$$F_X^{-1}(\xi) = \sqrt{2}erf^{-1}(2\xi - 1)$$

   For a standard normal.

Listing 4: Script for Problem

```
#!/usr/bin/env python3

#################################################################
##################### Import packages ##########################
#################################################################
import numpy as np
import time
import copy
import scipy.special as sps
import matplotlib.pyplot as plt
plt.rcParams["font.family"] = "monospace"
import matplotlib
matplotlib.rc('text',usetex=True)
matplotlib.rcParams['text.latex.preamble']=[r"\usepackage{amsmath}"]
start_time = time.time()
#################################################################
```

```
#################### Calculations ############################
##############################################################

fig=plt.figure()
ax=fig.add_subplot(111)
ax.set_xlabel(r'$\boldsymbol{x}$',fontsize=18)
ax.set_ylabel(r'\textbf{y}',fontsize=18)
ax.grid(alpha=0.8,color='black',linestyle='dotted')

B=3.826659

for i in range(0,10):

    l1=np.random.uniform(0,1)
    l2=np.random.uniform(0,1)

    U=(1-l2)**B
    Us=(1-l2)**(B-1)

    error=100

    count=0
    V=(1-l1)**B
    while(error>0.001):
        A=(V-U*(V-1))**((1/B)-1)
        C=l1/Us
        Vn=1-C/A
        error=abs(V-Vn)/Vn
        V=copy.copy(Vn)

        if count>999:
            print("Did not converge after 1000")
            quit()
        count=count+1
    W=1-V**(1/B)
    print(W)
    x=(2**0.5)*sps.erfinv(2*l1-1)
    y=(2**0.5)*sps.erfinv(2*W-1)
    ax.plot(x,y,'ko',markersize=5)

plt.savefig('P5.pdf')
```


1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
y
−3
−2
−1
0
1
2
3
4
x