HW6_Code

March 6, 2023

[ ]:
```
import random

from qutip import *
from qiskit.visualization import array_to_latex
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
from scipy.optimize import curve_fit
```

0.1 1.D)

[ ]:
```
def P(t):
    tau = 1
    return 1/tau * np.exp(-t/tau)
```

0.1.1 1.D.1) list of exponential distribution values where $\tau = 1$

[ ]:
```
samples = np.random.exponential(1, 1000)
samples
```

[ ]:
```
array([2.47540013e+00, 3.87756461e-01, 8.83609546e-01, 5.70115172e-01,
       6.76793915e-01, 1.73457164e+00, 2.85625778e+00, 4.30320783e-01,
       8.61177793e-01, 1.03874373e+00, 6.86943424e-02, 4.98476839e-01,
       7.04159314e-01, 1.02663654e-01, 1.54902430e+00, 3.48929469e+00,
       1.87133542e+00, 3.88035149e+00, 1.00374327e+00, 1.76165115e+00,
       7.58613472e-01, 3.20901794e+00, 3.10467048e-01, 2.85622658e-01,
       4.74446759e-02, 1.82727913e+00, 1.21133824e+00, 2.27022001e+00,
       2.25555167e+00, 9.59208619e-01, 2.19369326e-01, 1.84850007e-01,
       3.00263350e+00, 4.91835042e-01, 6.41950902e-01, 7.89352999e-01,
       7.26564322e-01, 2.15268902e+00, 9.75831948e-01, 1.67316804e+00,
       6.62696027e-01, 2.21591293e+00, 2.79193810e+00, 4.43373935e-01,
       1.98224648e+00, 1.24254198e+00, 8.82640135e-01, 8.02800794e-01,
       3.85072564e+00, 2.85109132e-01, 1.00523902e+00, 2.62196023e+00,
       9.63390588e-01, 2.29528553e-01, 8.28382631e-02, 5.30200806e-02,
       5.40839682e-02, 3.59113423e-01, 2.58620554e-01, 1.32230267e+00,
       5.30811106e-01, 1.45636722e+00, 1.07947272e+00, 9.83926755e-01,
```

1.08997435e-01, 8.73328278e-01, 1.28053429e+00, 6.36646454e-02,
5.36091591e-01, 4.01514231e-01, 4.67678694e-01, 8.15274238e-01,
4.65693041e-01, 2.02132104e+00, 2.62543355e+00, 1.77728322e+00,
1.11340712e-01, 2.57743091e+00, 4.92499325e-02, 7.32563827e-01,
1.75680487e+00, 1.32916188e+00, 2.69371834e-01, 6.90992542e-02,
9.73108330e-01, 1.83457440e-01, 2.75491560e+00, 2.85757471e-01,
8.84007851e-02, 8.57711928e-01, 7.74637473e-01, 8.93173825e-01,
1.28696556e-01, 6.87805524e-01, 4.35457039e-01, 6.86444338e-01,
1.59753000e+00, 2.74655564e+00, 2.90100091e+00, 1.79790422e+00,
5.67784119e-01, 6.87724815e-01, 1.17910389e+00, 1.64321080e+00,
1.00430230e+00, 9.18152991e-01, 2.18491885e+00, 6.09149539e-01,
6.01369471e-01, 1.11449739e+00, 7.56380903e-01, 1.91670064e-01,
2.01529080e-01, 1.99754254e+00, 3.99215149e-01, 8.69803259e-01,
1.60688099e+00, 1.25048576e+00, 5.46426168e-02, 3.81549673e+00,
6.69769598e-01, 4.43194437e-01, 1.38010855e-02, 7.54354816e-01,
5.34371489e-01, 8.43795970e-01, 1.14288208e+00, 7.99949494e-03,
8.17551079e-01, 1.62860151e+00, 1.51764193e+00, 3.55732059e-01,
1.25628916e+00, 1.45576466e+00, 1.04227892e-02, 8.51411951e-01,
1.09693614e+00, 8.83675533e-01, 1.86089032e+00, 2.79180974e-03,
1.24973360e+00, 6.36161364e-01, 5.66955809e-01, 2.76770445e+00,
9.20708784e-02, 9.25330483e-01, 9.05826153e-02, 5.32767313e-01,
2.15724582e-01, 1.69999672e-01, 8.49639733e-01, 4.21186270e-01,
6.18105320e-01, 2.33967658e+00, 4.56151344e-02, 5.61051517e-01,
2.04111097e-01, 2.50430580e-01, 7.32858733e-02, 1.35856596e+00,
2.36707654e-03, 1.68164852e-01, 2.94011331e-02, 2.39421116e-01,
2.32181351e+00, 5.45051914e-01, 1.06056132e-01, 3.25530332e-01,
3.88836744e-01, 1.21849458e-01, 6.18941530e-01, 9.00115631e-02,
5.74644937e-02, 6.82703225e-01, 8.32138594e-02, 9.11015735e-01,
1.23868529e+00, 4.06715909e-01, 3.40237353e-01, 1.04027421e+00,
6.43540632e-01, 1.97083826e+00, 2.92002201e-01, 1.66498962e-01,
1.04558409e+00, 2.31537012e+00, 3.75055580e-01, 4.35296459e-01,
3.81649863e-02, 3.09653913e-01, 2.45261463e+00, 5.78318895e-01,
2.41958286e-01, 1.39214991e+00, 5.64179233e-01, 1.63551565e+00,
3.64127573e-02, 2.66742681e+00, 6.71602966e-01, 1.93902848e+00,
1.36136086e+00, 1.73439870e-01, 1.31331537e+00, 2.13781416e-01,
3.78948079e-01, 1.03587818e+00, 1.93126151e+00, 2.46118366e+00,
1.19464294e-01, 1.07571208e+00, 4.70926891e-01, 5.04551377e-01,
1.73297845e-01, 5.81352978e-01, 3.89273869e-01, 3.14577715e+00,
2.40768673e-01, 1.19245354e+00, 3.53223666e+00, 1.04424376e+00,
1.88647132e+00, 8.13431270e-01, 9.87891961e-01, 2.50377782e-01,
1.80348251e-02, 3.24785446e+00, 3.98003734e-01, 3.16326877e-02,
8.03828697e-01, 1.47556124e+00, 1.19507607e+00, 8.21759613e-01,
2.70612956e+00, 6.36226429e-01, 2.04653199e+00, 1.79764521e-01,
2.25023343e-01, 1.79502683e-01, 1.17639569e+00, 2.14764285e+00,
1.85393453e-01, 7.06077753e-01, 2.73563276e+00, 1.43268251e+00,
6.31512417e-02, 6.18935495e-01, 1.13065169e+00, 3.91730518e-01,
2.23532188e+00, 1.31567114e+00, 3.44405462e-01, 1.27475020e-02,

7.64576852e-01, 7.92838715e-01, 1.54994581e-01, 3.96213070e-01,
1.98306322e-01, 3.37603719e-01, 1.37075620e+00, 2.09266764e-01,
2.60614875e+00, 1.26639010e+00, 3.49446800e+00, 1.64955022e+00,
4.27063022e-01, 5.45652502e-01, 4.57783135e-01, 2.63296965e+00,
5.89771768e-01, 5.27895969e-01, 2.35481681e+00, 1.51330848e-01,
1.11006997e+00, 5.61683099e-01, 1.81432628e+00, 2.72546045e+00,
1.30591523e+00, 1.20925128e+00, 9.75470904e-01, 5.50452830e-01,
2.60318059e-01, 3.98358484e+00, 7.11081044e-01, 9.90871776e-01,
5.28335358e+00, 1.13396879e+00, 6.05142369e-01, 5.99877438e-01,
4.23881763e-01, 4.30710468e-01, 6.71969727e-01, 7.20279477e-02,
1.07757272e-01, 8.77777340e-02, 3.42665629e+00, 4.03296393e-01,
6.57321740e-03, 1.83824881e-01, 5.77035958e-01, 3.60782162e-01,
6.46177136e-01, 1.58619584e+00, 5.03528723e-01, 2.19294371e+00,
4.59897498e-01, 1.37263002e+00, 2.65675801e-01, 1.60697377e-01,
8.78541295e-01, 2.56674626e-01, 3.56054088e-02, 1.96725066e-02,
8.31313500e-01, 3.94949074e-01, 4.13894828e-01, 1.12149591e+00,
1.22595017e+00, 1.81322563e-02, 2.50388534e+00, 1.32804689e+00,
5.98996232e-02, 2.15177556e-01, 1.57264989e+00, 7.13748595e-01,
1.36442993e+00, 2.00601947e-01, 2.60699368e+00, 2.14332321e-01,
3.42107609e-01, 3.11557854e-01, 1.74862975e+00, 2.09207891e+00,
6.15326229e-01, 6.32544057e-01, 1.28032830e+00, 1.05970829e+00,
6.71416160e-01, 2.14767548e-01, 1.44428824e+00, 1.35997737e+00,
6.16087001e-02, 1.37115255e+00, 1.44900063e+00, 9.25835853e-02,
1.62189048e+00, 4.21051894e-01, 7.41437094e-01, 2.37176886e-02,
6.08188059e-01, 8.62222984e-01, 4.83004397e-02, 7.01945051e-01,
1.52115879e+00, 4.21398170e-01, 2.73233604e+00, 2.54809258e+00,
1.29460326e+00, 1.99208999e+00, 2.38781259e-02, 6.58704353e-01,
2.68774358e+00, 1.52314308e-01, 1.50344635e-01, 2.90973844e-01,
1.30890633e+00, 4.99361152e-01, 1.30177025e+00, 9.50236429e-01,
1.58178237e+00, 1.86486141e+00, 1.29509647e+00, 3.74846331e-01,
1.06789965e+00, 5.07017535e-01, 4.63449957e-01, 2.25895196e+00,
1.77063223e+00, 1.63197100e-01, 8.07170877e-01, 5.06346876e-01,
6.96865237e-01, 6.35112923e-02, 8.37770002e-01, 2.79290787e-01,
2.58001186e+00, 1.47581223e+00, 1.60277958e+00, 4.36945150e-01,
2.27078378e+00, 8.90494777e-01, 2.15960523e+00, 5.81475545e-01,
7.77261395e-01, 3.82369776e-01, 4.57718385e-01, 1.62616971e+00,
7.75462265e-02, 4.60363551e+00, 7.63410716e-02, 3.24551795e+00,
1.22180613e+00, 8.85726578e-01, 2.36444280e+00, 1.37666043e+00,
2.10769664e+00, 2.63771913e-02, 1.54902156e+00, 3.27375393e+00,
2.71463634e+00, 2.10283402e-01, 9.56558934e-02, 2.98827016e+00,
1.40299605e-01, 1.03416805e+00, 3.45693878e-01, 4.66704603e-01,
1.39910099e+00, 9.80984485e-01, 2.07522821e+00, 1.13017444e+00,
1.01662731e+00, 2.73209389e-01, 3.21023153e+00, 1.98218515e-04,
3.12353978e+00, 7.22749177e-01, 1.45358484e+00, 1.20818567e+00,
7.50461707e-03, 8.40661912e-01, 1.61358119e+00, 7.49857105e-01,
7.80148562e-01, 4.62199063e-01, 3.17852032e-01, 1.67838819e+00,
9.62612664e-01, 4.76267875e-02, 4.79544477e-01, 4.04789026e+00,

2.97330971e-01, 7.60206550e-02, 1.17146706e+00, 4.33186732e-01,
5.47962994e-02, 3.94431219e-02, 3.44479090e-01, 4.58924288e-02,
7.22693418e-01, 1.37941386e+00, 9.05006354e-01, 1.90642717e+00,
2.12817490e-01, 1.07834596e+00, 1.43223691e+00, 1.47002005e+00,
6.70317835e-01, 4.35985925e-02, 7.03346994e-01, 9.20257946e-02,
1.17974710e+00, 1.89743274e-01, 2.36522307e+00, 4.31666694e+00,
2.19403351e+00, 8.05084185e-01, 3.10575121e-01, 2.54564342e-01,
1.58483418e+00, 9.38284379e-01, 1.21480016e+00, 3.19651323e-01,
8.26053232e-01, 1.45114664e+00, 1.22286507e+00, 2.96224687e-04,
1.79387050e+00, 1.25351208e+00, 1.05005534e+00, 1.11183725e+00,
3.91059648e-01, 4.62878490e-01, 7.60785361e-01, 4.41829878e-01,
2.30754389e-02, 1.21359919e+00, 2.68502135e+00, 9.17063407e-03,
2.30642460e+00, 1.28008904e-01, 2.14351006e+00, 1.35345071e+00,
4.98612298e-01, 4.44353503e-01, 5.30391012e-01, 3.56801819e-01,
8.84638216e-01, 1.84322569e+00, 7.20142474e-02, 3.63339487e-02,
4.90807961e-01, 9.76434344e-01, 2.07240534e+00, 6.87126928e-02,
7.43034403e-01, 1.07180555e+00, 1.14408899e-02, 3.85017637e-02,
1.86998555e+00, 2.94103638e-02, 2.45308427e-01, 6.18731929e-01,
4.69299830e+00, 2.90864984e-01, 2.43707686e-02, 1.90276198e+00,
2.40762737e+00, 4.88607979e-01, 1.67892461e-01, 1.93462855e+00,
1.74631161e+00, 5.34683872e-01, 3.57536488e-01, 6.19472901e-01,
2.50706042e+00, 1.08921107e+00, 1.05381115e+00, 1.48457546e-01,
3.96333488e-01, 8.65911459e-01, 4.30475404e-02, 9.53303744e-01,
9.55463042e-01, 2.13644961e+00, 1.35272785e+00, 2.01264241e+00,
6.05411382e-01, 5.95413810e-01, 3.96112815e-01, 3.38926018e-01,
8.63132193e-01, 3.01123577e-01, 9.42647837e-01, 1.04583656e+00,
1.01080299e+00, 1.04082119e+00, 1.19236506e-01, 5.40759514e-01,
4.49462898e-01, 1.38420546e+00, 3.24767582e+00, 1.59274648e+00,
1.17575622e+00, 2.76639195e-01, 1.33059089e+00, 1.97195372e+00,
1.04588904e+00, 5.16744175e-02, 5.49580518e-01, 1.10438534e+00,
3.50961076e+00, 1.31867130e-01, 4.28440943e-01, 3.10313395e+00,
9.49388281e-01, 1.24364915e+00, 2.82226298e+00, 4.35660776e-01,
1.49129688e+00, 8.64747956e-01, 1.53749681e+00, 1.14534896e+00,
3.81678523e+00, 6.08852695e-01, 9.23223811e-02, 8.83832260e-02,
5.57476404e+00, 5.15518633e-01, 1.55631471e+00, 1.22322463e-01,
9.79041302e-01, 7.01264495e-01, 2.91311028e+00, 9.18698331e-01,
5.23741916e-01, 5.12337366e-01, 2.10931032e+00, 3.31031155e-02,
9.48212535e-02, 3.09811818e+00, 2.71393293e-02, 8.73787754e-01,
9.71510341e-01, 2.76728416e-01, 1.73074296e+00, 2.32269253e+00,
1.60051829e+00, 7.84095838e-01, 1.64986868e-01, 4.85617839e-01,
6.41017595e-01, 3.03845840e-01, 9.10352419e-01, 3.01496374e-01,
2.00261722e+00, 1.35847006e-01, 2.25183057e+00, 1.79820376e-01,
3.36889874e-02, 1.46976065e-01, 2.66619358e-01, 1.23176187e+00,
1.07469449e+00, 1.49914667e+00, 9.24620783e-01, 1.76464994e+00,
4.13242087e-01, 2.15943491e-01, 1.32051131e+00, 1.13975128e-01,
2.42902024e+00, 1.56797149e+00, 8.57225458e-01, 1.95455034e+00,
4.14272768e-01, 6.35349266e-02, 9.15316505e-01, 1.27572796e+00,

1.53059958e+00, 5.23246597e-01, 1.35205172e-01, 4.63514661e-01,
6.45494594e-01, 2.25153808e+00, 3.02569548e+00, 1.14825544e+00,
3.24375619e-01, 4.95942664e-01, 2.45896815e-01, 1.49995406e-01,
6.58282842e-01, 1.66641811e+00, 2.87390793e-01, 7.71472815e-01,
1.10642403e-02, 2.19291030e-01, 2.96739859e-01, 1.45771433e+00,
2.15045610e-01, 1.84646965e+00, 3.09277544e-01, 1.61778383e+00,
4.76667656e-01, 2.04889837e-01, 1.63913121e-01, 2.37859652e-01,
4.66352583e-01, 6.76664783e-01, 2.24090253e-01, 1.06160976e+00,
8.49164844e-01, 3.43229372e+00, 1.06018852e-01, 4.85597309e-03,
1.08028672e+00, 8.37711789e-01, 1.32430216e+00, 1.08140374e+00,
2.02579540e+00, 7.69005900e-01, 5.89650876e-01, 1.63547315e+00,
1.30910583e+00, 1.71278942e-01, 1.39347989e+00, 5.73712188e-01,
2.69960500e-01, 1.49469457e-01, 1.90639868e+00, 2.09976178e-01,
9.25688106e-01, 2.90608483e-01, 7.15705903e-01, 1.14878673e+00,
2.43493866e-01, 7.23422807e-01, 7.87542824e-01, 6.95733376e-01,
8.70077648e-01, 7.77512434e-01, 1.98419174e-01, 4.73297694e-02,
1.66318355e-01, 2.33484082e+00, 1.66828874e-01, 4.71491426e-01,
6.12504804e-01, 5.44130704e-01, 6.10921956e-01, 1.10453739e-01,
1.07111288e+00, 4.12652986e-01, 9.92158355e-01, 5.78298687e-01,
1.22208578e+00, 4.41125970e-01, 1.17513840e+00, 2.76824924e+00,
1.86975520e-01, 8.76973603e-02, 7.28120308e-01, 1.14028153e+00,
8.79561880e-01, 1.74336388e+00, 1.17089697e+00, 1.55993538e-01,
1.49996983e-02, 4.67196802e-01, 3.93749415e-01, 1.80596988e+00,
1.02449628e-01, 3.28447671e+00, 3.31807347e-02, 3.05189744e-01,
8.92965942e-01, 3.67783821e-01, 2.34912713e-01, 6.75079519e-01,
1.79633092e-01, 3.49939434e-01, 1.38716238e+00, 7.78709416e-01,
1.18644733e+00, 1.75849050e+00, 3.59664983e+00, 1.03726704e+00,
4.54443126e-01, 9.41757922e-01, 5.71155293e-01, 2.04326273e-01,
2.20569637e+00, 7.97965732e-01, 1.10961213e-01, 4.91621599e-02,
9.52903761e-01, 1.78915160e+00, 2.70095207e+00, 1.90848387e-01,
1.25917807e+00, 3.80965060e-01, 1.28837950e+00, 1.40764498e+00,
6.12511122e-01, 3.80625220e-01, 3.96574036e+00, 1.81807026e+00,
1.32317747e-01, 1.85249718e+00, 1.71361368e-01, 2.96870047e-01,
5.84258906e-01, 3.30605379e-02, 1.10850240e+00, 1.35768515e+00,
1.72480319e+00, 3.00437356e-01, 7.31846379e-01, 1.01990670e+00,
1.14879876e-01, 1.99467380e+00, 3.08023430e+00, 1.46371065e+00,
1.55070513e+00, 2.13948356e+00, 3.89150376e+00, 4.64998527e-01,
7.95010838e-01, 1.01558174e+00, 1.80328846e+00, 3.48732763e+00,
3.72223754e+00, 1.57900038e+00, 1.18603170e-01, 4.20709505e+00,
3.83987029e-01, 2.25920481e-01, 1.52903717e-01, 3.12404907e-01,
4.05554990e+00, 9.14875670e-01, 1.63966077e+00, 1.10571325e+00,
9.94140395e-02, 9.79886601e-01, 1.06328291e+00, 7.81863095e-02,
1.39750386e+00, 2.26460505e+00, 1.87438958e+00, 2.94995535e-02,
4.81764038e+00, 1.50837552e+00, 2.75692559e-01, 1.21153087e+00,
2.29935886e+00, 1.04953747e+00, 6.90689851e-01, 2.28974176e+00,
1.32026202e+00, 4.57679966e-01, 3.81448721e-01, 7.25123936e-01,
1.83286171e-01, 4.42662221e-01, 1.27032826e+00, 7.36564848e-01,

```
3.19380629e-01, 1.87839853e+00, 2.58842142e-02, 1.51804444e+00,
2.07910160e+00, 3.82938326e-01, 7.43523794e-01, 1.37142251e+00,
1.09455789e+00, 2.21896628e-01, 8.41585336e-01, 1.32027835e+00,
4.76598599e+00, 9.38186739e-01, 1.24562622e+00, 1.62209757e+00,
8.41220612e-01, 1.93015225e+00, 7.10351242e-02, 5.80168993e-01,
3.59086733e-01, 1.16101119e-01, 2.44405329e+00, 2.34246423e-01,
1.12439369e+00, 2.75659750e-01, 6.66484820e-01, 1.28241364e+00,
1.40903164e+00, 7.94546910e-02, 6.27720867e-01, 1.22474218e+00,
2.93624563e+00, 5.18663602e-01, 3.27381187e+00, 7.91590700e-01,
6.25634386e-02, 1.65781124e+00, 4.41770105e-01, 1.99266385e-01,
4.73345538e+00, 3.34483139e-01, 9.55530195e-01, 1.01549904e+00,
4.46525165e-02, 1.86172650e-01, 9.99322268e-02, 1.50899070e-01,
1.11154526e+00, 1.59235636e+00, 2.58522941e+00, 7.58276318e-01,
1.23107489e+00, 6.10121188e-02, 1.27979892e+00, 4.66537763e-01,
2.17009519e-02, 5.72768179e-02, 2.49467684e+00, 2.95096705e-01,
1.99013851e+00, 9.65223164e-02, 3.68847534e-01, 6.18020015e-01,
2.02780745e+00, 4.58047585e-01, 4.46059142e-01, 1.67786028e-02,
1.28513827e+00, 3.02829810e+00, 2.75031431e+00, 5.22976342e-01,
4.57249556e-01, 2.58300248e-01, 1.93403439e+00, 1.98239204e-01,
1.85900787e+00, 3.00271998e+00, 9.52541143e-02, 3.48349331e-01,
2.87049915e+00, 3.89329938e-01, 2.60118434e-01, 9.68333567e-01,
5.13455468e-01, 2.31507386e+00, 6.43397483e-01, 4.25313680e-01,
3.41859302e-01, 1.54598584e+00, 8.16374654e-01, 1.03634274e+00,
1.48692066e-01, 3.17547069e-01, 5.04786479e-01, 1.52208203e+00,
1.09591923e+00, 1.86162196e+00, 3.73560643e+00, 5.63759307e-01,
1.50931787e+00, 1.09437522e+00, 8.35082081e-01, 4.38727361e-01,
3.62961194e+00, 5.22770413e-01, 5.96411810e-01, 1.87960806e+00,
6.11373212e-01, 1.27587348e+00, 8.67040711e-01, 4.90216769e-01,
3.92308510e-01, 1.08031589e+00, 5.42979472e-01, 1.38468510e+00,
6.16801387e-02, 1.13737780e+00, 1.22108799e-01, 8.08265981e-01,
2.73490549e+00, 1.74380581e-01, 2.11937964e-01, 3.05406739e-01,
3.71207914e-01, 1.52789809e+00, 1.16240956e+00, 1.20047741e+00,
6.00148761e-01, 1.32569989e-01, 1.31567854e+00, 5.51026628e-01,
1.16435834e+00, 2.22050179e-01, 4.29071658e+00, 1.07428935e+00,
1.19164063e-01, 2.32610178e+00, 6.91797544e-01, 1.21400750e-01,
1.50996006e+00, 1.10561320e+00, 2.45520987e-01, 1.31639241e-01,
3.44700524e-01, 3.26935073e-01, 1.76378983e+00, 1.32372660e+00,
1.55650235e-01, 6.24924995e-01, 5.36583207e-01, 1.77591631e+00,
7.65925964e-01, 2.46720075e+00, 2.05422147e+00, 2.40413969e+00,
1.61445073e+00, 4.86059445e+00, 1.70096818e+00, 3.95563902e-02,
1.49178787e-01, 8.99399122e-02, 8.20915946e-01, 1.06601431e-01,
4.43626488e-01, 2.23777258e-01, 6.15369018e-01, 6.15904299e-01,
1.19049256e-01, 1.57703916e-01, 2.58549167e+00, 1.62848832e+00,
1.00044321e+00, 3.54972907e-01, 2.52789234e-01, 6.22627221e-01,
7.63523536e-01, 5.04549720e-01, 7.63509206e-01, 8.80977010e-01,
2.56594149e+00, 2.46639403e+00, 6.69337849e-01, 3.89069078e-01])
```

0.1.2 1.D.2) Mean of the list

[ ]:
```python
print(np.mean(samples))
```

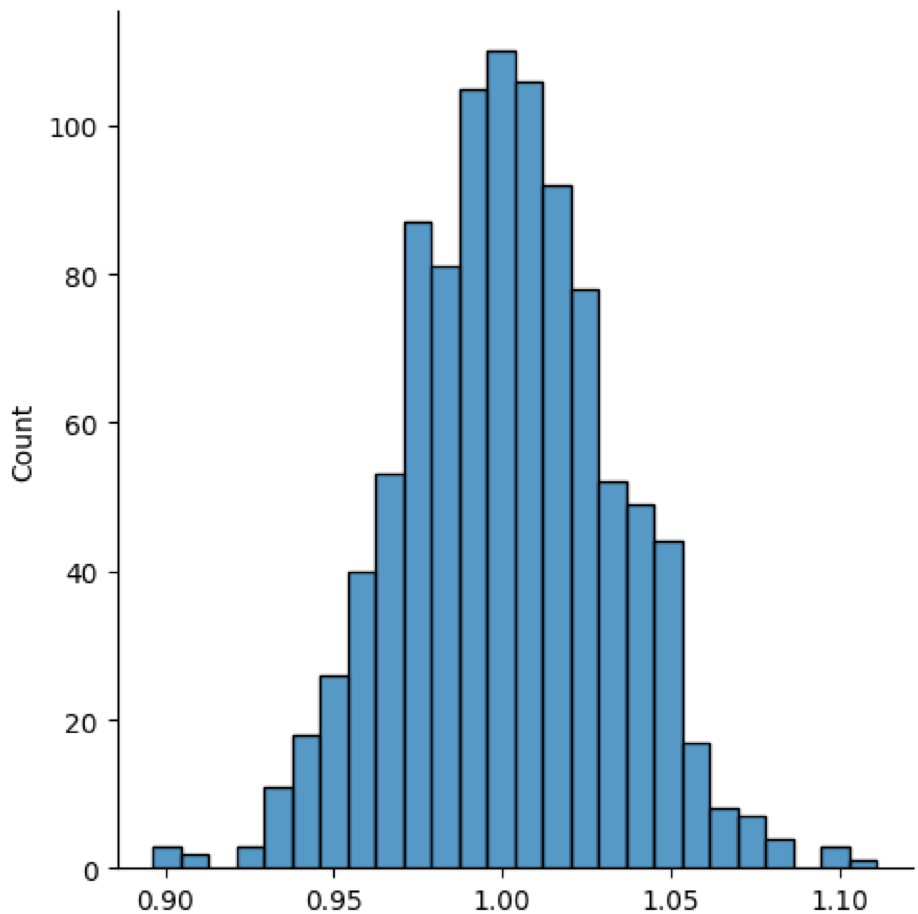
```
1.0169658369496997
```

[ ]:
```python
mean_dsit = []

for i in range(0,1000):
    samples = np.random.exponential(1, 1000)
    mean = np.mean(samples)

    mean_dsit.append(mean)

sns.displot(mean_dsit)
plt.show()
```



We can see that the above distribution is effectively Gaussian. This is a simulation of what our Likiness function would be for the $\tau_e = 1$

0.2 2)

0.2.1 2.A)

First, lets read these files into pandas dataframes like so:

```
[ ]: state_temp_df = pd.read_csv('./climdiv_state_year.csv')
     state_temp_df
```

```
[ ]:       fips  year       temp      tempc
     0        1  1895  61.641667  16.467593
     1        1  1896  64.266667  17.925926
     2        1  1897  64.191667  17.884259
     3        1  1898  62.983333  17.212963
     4        1  1899  63.100000  17.277778
     …     …    …          …          …
     5995    56  2015  44.158333   6.754630
     5996    56  2016  43.908333   6.615741
     5997    56  2017  43.200000   6.222222
     5998    56  2018  42.408333   5.782407
     5999    56  2019  40.383333   4.657407

     [6000 rows x 4 columns]
```

```
[ ]: state_num_df = pd.read_csv('./noaastate.txt')
     state_num_df
```

```
[ ]:     noaa_state_order           state
     0                  1         Alabama
     1                  2         Arizona
     2                  3        Arkansas
     3                  4      California
     4                  5        Colorado
     5                  6     Connecticut
     6                  7        Delaware
     7                  8         Florida
     8                  9         Georgia
     9                 10           Idaho
     10                11        Illinois
     11                12         Indiana
     12                13            Iowa
     13                14          Kansas
     14                15        Kentucky
     15                16       Louisiana
     16                17           Maine
     17                18        Maryland
     18                19   Massachusetts
     19                20        Michigan
```

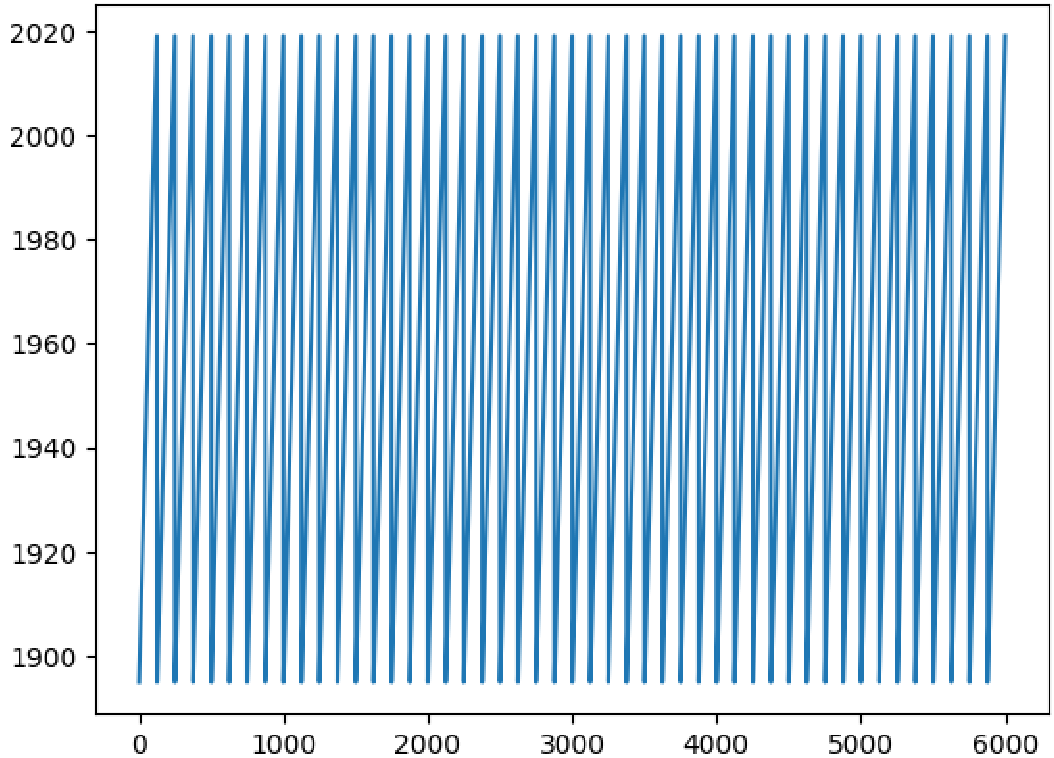
```
20          21        Minnesota
21          22      Mississippi
22          23         Missouri
23          24          Montana
24          25         Nebraska
25          26           Nevada
26          27    New Hampshire
27          28       New Jersey
28          29       New Mexico
29          30         New York
30          31   North Carolina
31          32     North Dakota
32          33             Ohio
33          34         Oklahoma
34          35           Oregon
35          36     Pennsylvania
36          37     Rhode Island
37          38   South Carolina
38          39     South Dakota
39          40        Tennessee
40          41            Texas
41          42             Utah
42          43          Vermont
43          44         Virginia
44          45       Washington
45          46    West Virginia
46          47        Wisconsin
47          48          Wyoming
```

0.2.2 2.B)

Lets take a look at the year data. As we can see, this year data oscillates back and forth. This is due to the fact that we're looking at the years for all 50 states - which are all labeled nicely in our `state_num_df` object

[ ]:
```
plt.plot(state_temp_df['year'])
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7ffa658b9b10>]
```



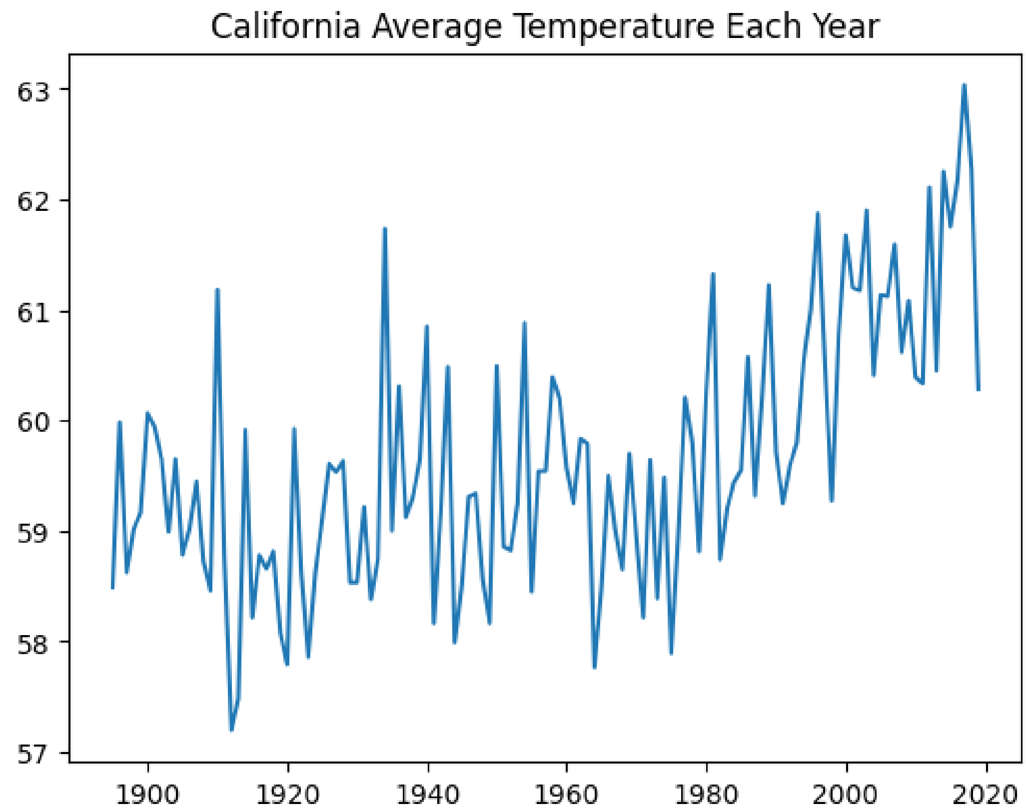
0.2.3 2.C)

[ ]:
```
ca_temp_df = state_temp_df.loc[state_temp_df['fips'] == 4]
ca_temp_df
```

[ ]:
```
     fips  year       temp      tempc
125     4  1895  58.491667  14.717593
126     4  1896  59.983333  15.546296
127     4  1897  58.625000  14.791667
128     4  1898  59.016667  15.009259
129     4  1899  59.166667  15.092593
..    …     …          …          …
245     4  2015  61.750000  16.527778
246     4  2016  62.175000  16.763889
247     4  2017  63.033333  17.240741
248     4  2018  62.266667  16.814815
249     4  2019  60.283333  15.712963

[125 rows x 4 columns]
```

[ ]:
```python
plt.plot(ca_temp_df['year'], ca_temp_df['temp'], label='California Temperature')
plt.title('California Average Temperature Each Year')
plt.show()
```



0.2.4 2.D)

[ ]:
```python
ny_temp_df = state_temp_df.loc[state_temp_df['fips'] == state_num_df.
 ↪loc[state_num_df['state'] == 'New York'].values[0][0]]
ny_temp_df
```

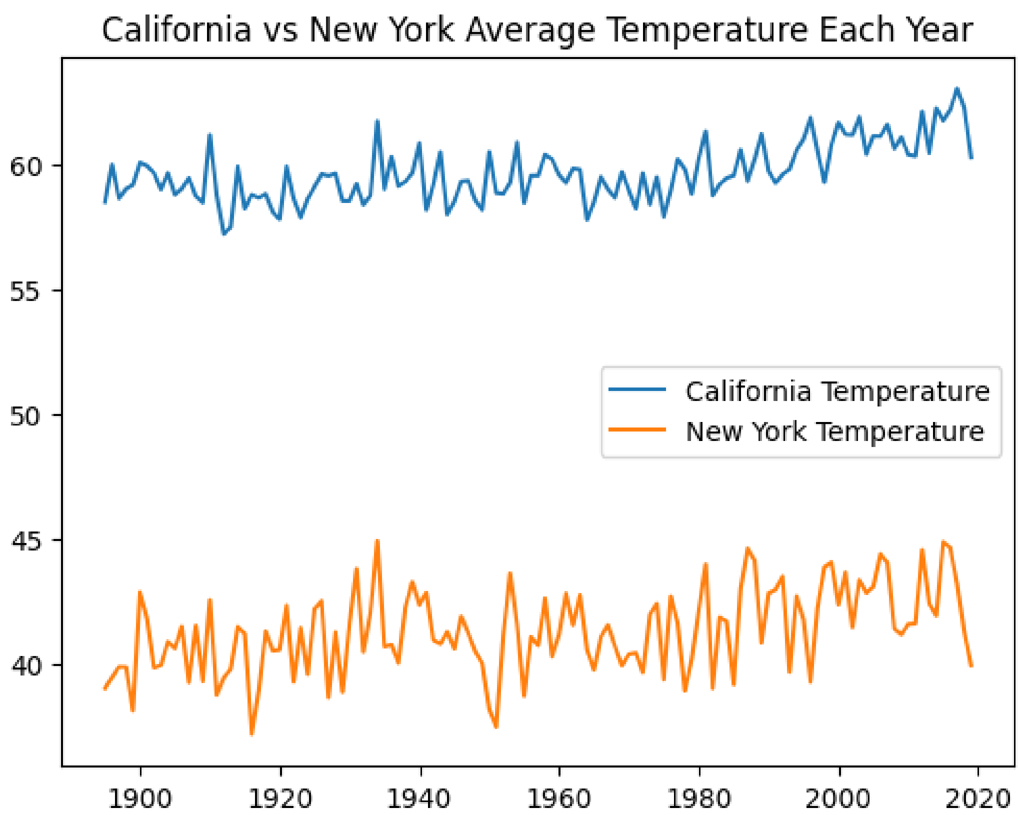
[ ]:

	fips	year	temp	tempc
2875	30	1895	38.991667	3.884259
2876	30	1896	39.433333	4.129630
2877	30	1897	39.850000	4.361111
2878	30	1898	39.841667	4.356481
2879	30	1899	38.100000	3.388889
…	…	…	…	…
2995	30	2015	44.858333	7.143519
2996	30	2016	44.625000	7.013889
2997	30	2017	43.175000	6.208333

```
2998    30  2018  41.275000  5.152778
2999    30  2019  39.908333  4.393519

[125 rows x 4 columns]
```

[ ]:
```
plt.plot(ca_temp_df['year'], ca_temp_df['temp'], label='California Temperature')
plt.plot(ny_temp_df['year'], ny_temp_df['temp'], label='New York Temperature')
plt.legend()
plt.title('California vs New York Average Temperature Each Year')
plt.show()
```



0.2.5 2.E)

[ ]:
```
total_mean_temp = state_temp_df['temp'].mean()
total_std_dev_temp = state_temp_df['temp'].std()
print(f"Total mean temperature is {total_mean_temp} and total standard␣
↪deviation is {total_std_dev_temp}")
```

```
Total mean temperature is 51.61530138888888 and total standard deviation is
8.005998486465774
```

0.2.6 2.F)

```
[ ]: ca_mean_temp = ca_temp_df["temp"].mean()
     ca_std_dev_temp = ca_temp_df["temp"].std()
     print(f"California's mean temperature is {ca_mean_temp} and standard deviation␣
      ↪is {ca_std_dev_temp}")

     ny_mean_temp = ny_temp_df["temp"].mean()
     ny_std_dev_temp = ny_temp_df["temp"].std()
     print(f"New York's mean temperature is {ny_mean_temp} and standard deviation is␣
      ↪{ny_std_dev_temp}")
```

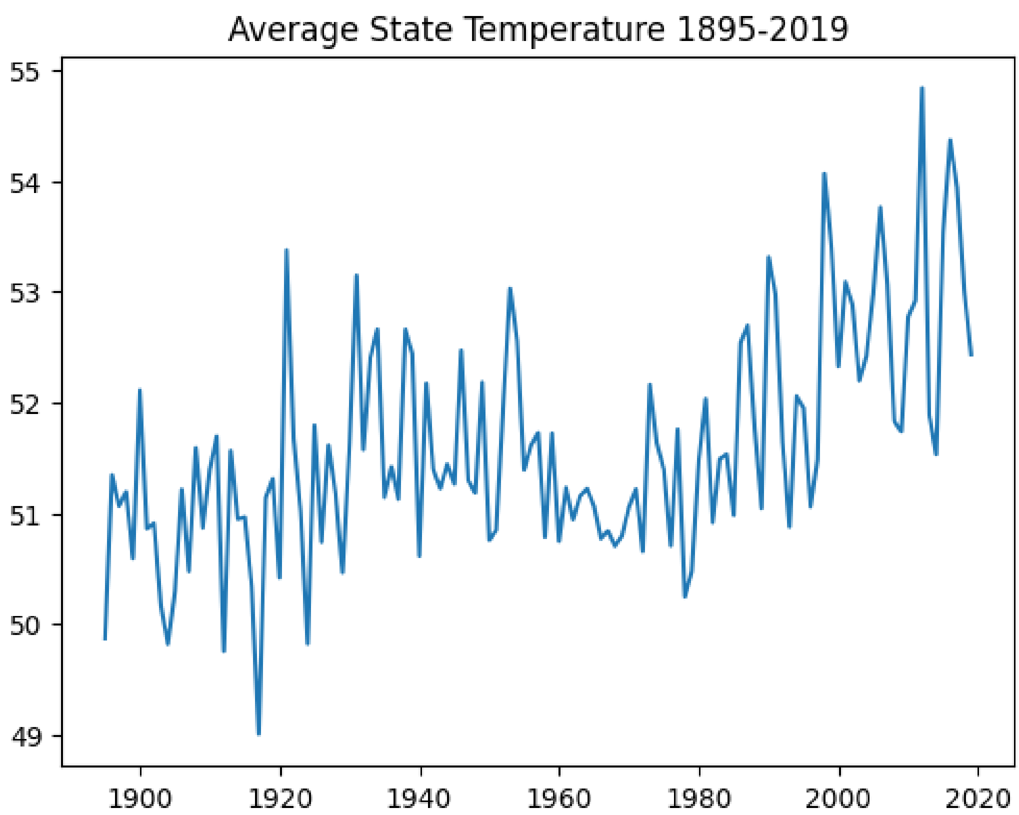
```
California's mean temperature is 59.65120000000001 and standard deviation is
1.1675177886175232
New York's mean temperature is 41.30486666666667 and standard deviation is
1.687916012706148
```

0.2.7 2.G)

```
[ ]: state_temp_1895_2019 = state_temp_df.loc[(state_temp_df['year'] >= 1895) &␣
      ↪(state_temp_df['year'] <= 2019)]


     state_mean_temps_1895_2019 = []
     for year in range(1895, 2020):
         state_mean_temps_1895_2019.append(state_temp_1895_2019.
      ↪loc[state_temp_1895_2019['year'] == year, 'temp'].mean())

     plt.plot(range(1895, 2020), state_mean_temps_1895_2019)
     plt.title('Average State Temperature 1895-2019')
     plt.show()
```



0.2.8 2.H)

[ ]:

```
total_mean_temp_first30years = state_temp_df.loc[state_temp_df['year'] <= min(state_temp_df['year']) + 30 ]['temp'].mean()
total_std_dev_temp_first30years = state_temp_df.loc[state_temp_df['year'] <= min(state_temp_df['year']) + 30 ]['temp'].std()
print(f"The total state mean temperature of the first 30 years was {total_mean_temp_first30years}")
print(f"The total sttate std deviation temperature of the first 30 years was {total_std_dev_temp_first30years}")

state_temp_df.loc[state_temp_df['year'] <= min(state_temp_df['year']) + 30 ]
```

```
The total state mean temperature of the first 30 years was 50.92079413082437
The total sttate std deviation temperature of the first 30 years was
8.265002842426076
```

[ ]:

	fips	year	temp	tempc
0	1	1895	61.641667	16.467593
1	1	1896	64.266667	17.925926

```
2          1  1897  64.191667  17.884259
3          1  1898  62.983333  17.212963
4          1  1899  63.100000  17.277778
…     …    …         …          …
5901      56  1921  42.225000   5.680556
5902      56  1922  39.116667   3.953704
5903      56  1923  39.558333   4.199074
5904      56  1924  38.708333   3.726852
5905      56  1925  41.500000   5.277778

[1488 rows x 4 columns]
```

0.2.9 2.I)

```
[ ]: total_mean_temp_last30years = state_temp_df.loc[state_temp_df['year'] >=
      ↪max(state_temp_df['year']) - 30 ]['temp'].mean()
     total_std_dev_temp_last30years = state_temp_df.loc[state_temp_df['year'] >=
      ↪max(state_temp_df['year']) - 30 ]['temp'].std()
     print(f"The total state mean temperature of the last 30 years was
      ↪{total_mean_temp_last30years}")
     print(f"The total sttate std deviation temperature of the last 30 years was
      ↪{total_std_dev_temp_last30years}")
```

```
The total state mean temperature of the last 30 years was 52.62523521505377
The total sttate std deviation temperature of the last 30 years was
7.8177710875692545
```

0.2.10 2.J)

```
[ ]: total_samples = 1488
     std_err = total_mean_temp_last30years / np.sqrt(total_samples)

     z_test = (total_mean_temp_first30years - total_mean_temp_last30years) / std_err
     print(f"The z-test value is {z_test}")
```

```
The z-test value is -1.2493652136148254
```

From our Z-test table this corresponds to a one-sided p value of 0.1056.

0.3 3)

0.3.1 3.A)

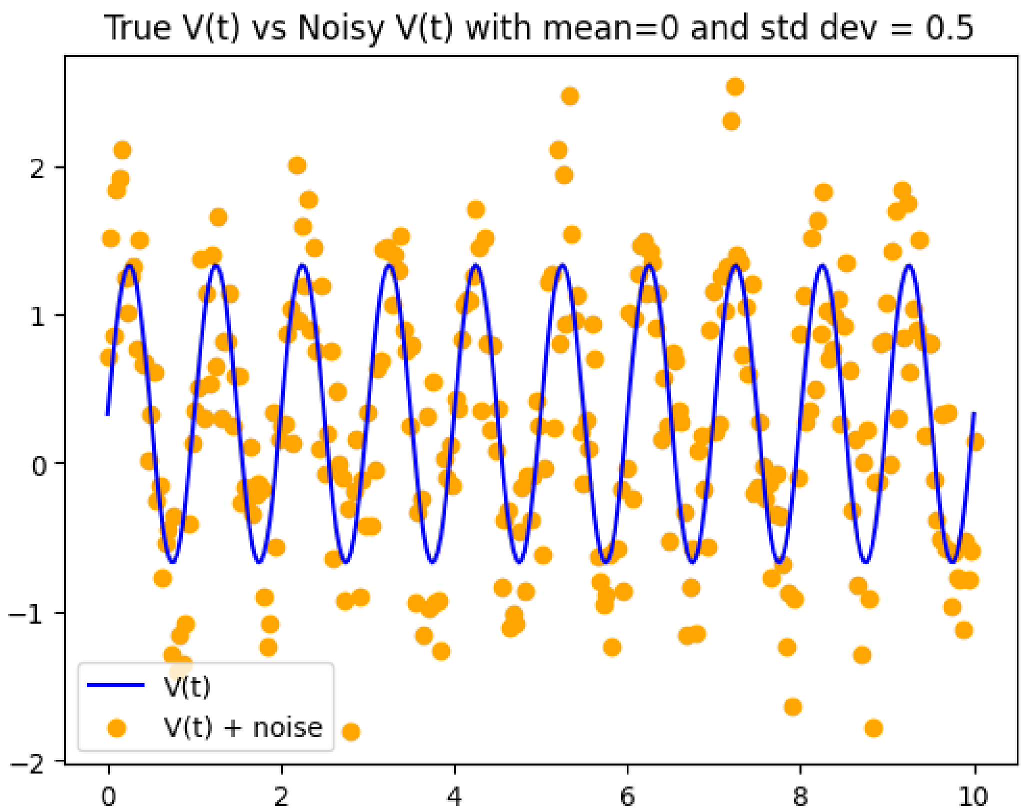
```
[ ]: def V(t):
         omega = 2*np.pi
         return 1 * np.sin(omega * t) + 0.33

     def V_sample(t):
```

```
    return V(t) + np.random.normal(0, 0.5, len(t))
```

[ ]:
```
num_samples = 300
t = np.linspace(0, 10, num_samples)

plt.title("True V(t) vs Noisy V(t) with mean=0 and std dev = 0.5")
plt.plot(t, V(t), label='V(t)', color='blue')
plt.scatter(t, V_sample(t), label='V(t) + noise', color='orange')
plt.legend(loc="lower left")
plt.show()
```



0.3.2 3.B)

[ ]:
```
# Curve fitted V
def V_B(t, B):
    omega = 2*np.pi
    return 1 * np.sin(omega * t) + B
```

[ ]:
```
popt, pcov = curve_fit(V_B, t, V_sample(t))
print(f"The curve fitted B value is {popt[0]}")
```

```
The curve fitted B value is 0.3379440445533258
```

As we can see, this fitted B value is very close to our actual value of 0.33!

0.3.3 3.C)

scipy curve fit is able to guess B because its sampling the difference in noise, which is gaussian by definition. The curve fit is effectively running a χ^2 test and minimizing the value to find the B value within a confidence interval. This works quite nicely for simple curves with gaussian-like noise, but breaks down for much more complicated curves and noise models.

0.3.4 3.D)

```
[ ]: def V_fit(t, A, B, f):
         return A * np.sin(2 *np.pi * f * t) + B
```

```
[ ]: popt, pcov = curve_fit(V_fit, t, V_sample(t), p0=[1, 0.33, 1])
     print(f"The curve fitted A value is {popt[0]}")
     print(f"The curve fitted B value is {popt[1]}")
     print(f"The curve fitted f value is {popt[2]}\n")

     print(f"Std dev of A is {np.sqrt(pcov[0][0])}")
     print(f"Std dev of B is {np.sqrt(pcov[1][1])}")
     print(f"Std dev of f is {np.sqrt(pcov[2][2])}\n")

     # 1.96 comes from the interval of 95% of the gaussian distribution
     print(f"Confidence interval for A is {popt[0] - 1.96 * np.sqrt(pcov[0][0])} to␣
      ↪{popt[0] + 1.96 * np.sqrt(pcov[0][0])}")
     print(f"Confidence interval for B is {popt[1] - 1.96 * np.sqrt(pcov[1][1])} to␣
      ↪{popt[1] + 1.96 * np.sqrt(pcov[1][1])}")
     print(f"Confidence interval for f is {popt[2] - 1.96 * np.sqrt(pcov[2][2])} to␣
      ↪{popt[2] + 1.96 * np.sqrt(pcov[2][2])}")
```

```
The curve fitted A value is 1.070895736399502
The curve fitted B value is 0.2871152177451105
The curve fitted f value is 1.0008456943753927

Std dev of A is 0.039302918377173895
Std dev of B is 0.027731738540511867
Std dev of f is 0.001004864283289341

Confidence interval for A is 0.9938620163802412 to 1.1479294564187628
Confidence interval for B is 0.23276101020570725 to 0.34146942528451374
Confidence interval for f is 0.9988761603801456 to 1.0028152283706397
```

Taking a look at these curve fitted values, we can see that they're very close to our actual values within a reasonable confidence interval

0.3.5 3.E)

[ ]:
```
array_to_latex(pcov, prefix="\\text{Covariance Matrix: }", precision=10)
```

[ ]:

$$\text{Covariance Matrix: } \begin{bmatrix} 0.0014368103 & 0 & 0.0000005122 \\ 0 & 0.000715694 & 0 \\ 0.0000005122 & 0 & 0.0000010193 \end{bmatrix}$$

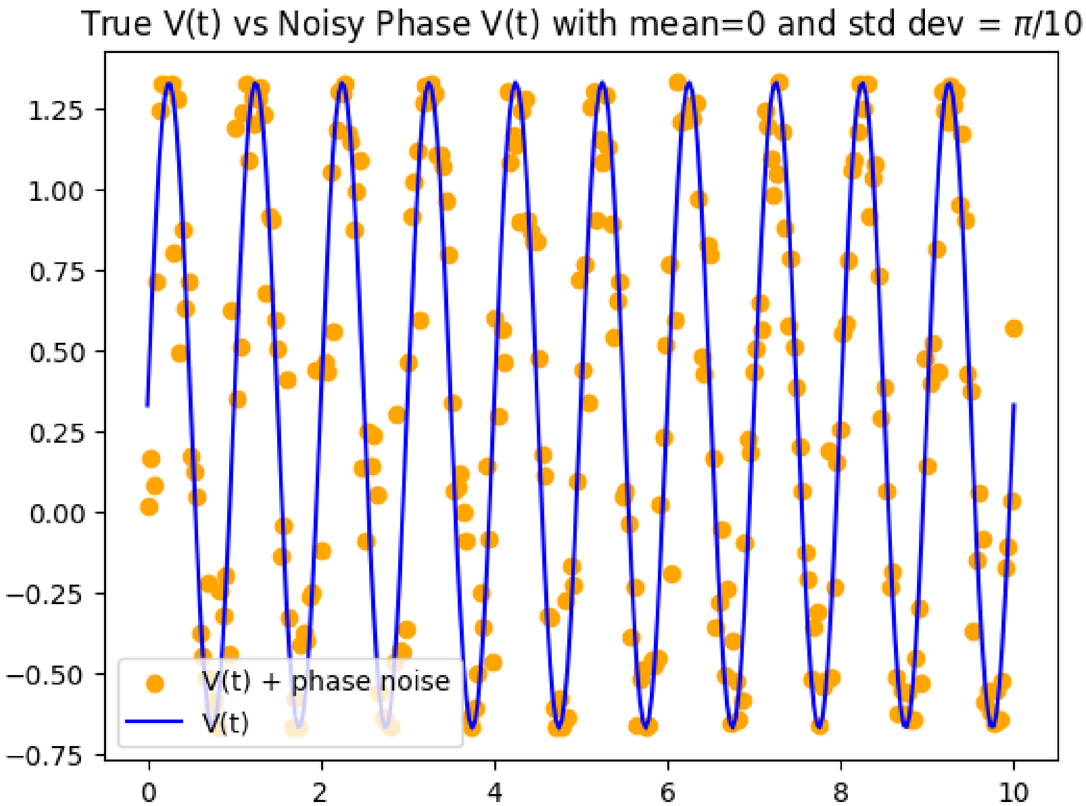
All of the elements represent the corresponding covariance of our A,B, and f parameters respectively. When the values are greater than zero, then they are more correlated. When they are less than zero, they are more anti-correlated.

However, our values are really close to zero. This implies they aren't really correlated with eachother at all.

0.3.6 3.F)

[ ]:
```
def V_sample_phase(t):
    omega = 2*np.pi
    return 1 * np.sin(omega * t + np.random.normal(0, np.pi/10, len(t))) + 0.33
```

[ ]:
```
plt.title("True V(t) vs Noisy Phase V(t) with mean=0 and std dev = $\pi/10$")
plt.scatter(t, V_sample_phase(t), label='V(t) + phase noise', color='orange')
plt.plot(t, V(t), label='V(t)', color='blue')
plt.legend(loc="lower left")
plt.show()
```




```
[ ]: popt, pcov = curve_fit(V_fit, t, V_sample_phase(t), p0=[1, 0.33, 1])
     print(f"The curve fitted A value is {popt[0]}")
     print(f"The curve fitted B value is {popt[1]}")
     print(f"The curve fitted f value is {popt[2]}\n")

     print(f"Std dev of A is {np.sqrt(pcov[0][0])}")
     print(f"Std dev of B is {np.sqrt(pcov[1][1])}")
     print(f"Std dev of f is {np.sqrt(pcov[2][2])}\n")

     # 1.96 comes from the interval of 95% of the gaussian distribution
     print(f"Confidence interval for A is {popt[0] - 1.96 * np.sqrt(pcov[0][0])} to␣
      ↪{popt[0] + 1.96 * np.sqrt(pcov[0][0])}")
     print(f"Confidence interval for B is {popt[1] - 1.96 * np.sqrt(pcov[1][1])} to␣
      ↪{popt[1] + 1.96 * np.sqrt(pcov[1][1])}")
     print(f"Confidence interval for f is {popt[2] - 1.96 * np.sqrt(pcov[2][2])} to␣
      ↪{popt[2] + 1.96 * np.sqrt(pcov[2][2])}")
```

```
The curve fitted A value is 0.9415789508365513
The curve fitted B value is 0.33089072866190444
The curve fitted f value is 1.0000710571310947
```

```
Std dev of A is 0.018555656843575116
Std dev of B is 0.013098057069055364
Std dev of f is 0.0005403974830958944

Confidence interval for A is 0.905208076702479 to 0.9779498249706235
Confidence interval for B is 0.3052185368065559 to 0.35656292051725297
Confidence interval for f is 0.9990118780642268 to 1.0011302361979626
```

[ ]:
```
array_to_latex(pcov, prefix="\\text{Covariance Matrix: }", precision=10)
```

[ ]:

$$\text{Covariance Matrix: } \begin{bmatrix} 0.0003443462 & 0 & \frac{0}{1} \\ 0 & 0.0001715591 & 0 \\ \frac{0}{1} & 0 & \frac{0}{1} \end{bmatrix}$$