MCMCS Summative assessment- 277227

```
import numpy as np
import matplotlib.pyplot as plt
```

Question 1

1)a)

```
#1)a)
def my_transf(A):
    # Ensure A is a NumPy array
    A = np.array(A)

    # Calculate the sum of the matrix and its transpose
    result = A + A.T

    return result

# Example usage:
# Define a matrix A
matrix_A = np.array([[1, 3, 5], [7, 9, 11], [13, 15, 17]])

# Call the function
result_matrix = my_transf(matrix_A)

# Display the result

print("Matrix A:")
print(matrix_A)

print("\nMatrix A + Aᵀ:")
print(result_matrix)
```

```
Matrix A:
[[ 1  3  5]
 [ 7  9 11]
 [13 15 17]]

Matrix A + Aᵀ:
[[ 2 10 18]
 [10 18 26]
 [18 26 34]]
```

1)b)

```
#1)b)
def dot_product_of_eigenvectors(input_matrix, index_i, index_j):
    # Ensure the input_matrix is a NumPy array
    input_matrix = np.array(input_matrix)

    # Calculate eigenvalues and eigenvectors
    eigenvalues, eigenvectors = np.linalg.eig(input_matrix)

    # Extract eigenvectors corresponding to indices index_i and
index_j
    eigenvector_i = eigenvectors[:, index_i]
    eigenvector_j = eigenvectors[:, index_j]

    # Calculate dot product of the two eigenvectors
    dot_product_result = np.dot(eigenvector_i, eigenvector_j)

    return dot_product_result

# Choose indices new_index_i and new_index_j (ensure they are valid
indices for the matrix size)
new_index_i = 0
new_index_j = 1

# Call the function with the new matrix and indices
result_new = dot_product_of_eigenvectors(matrix_A, new_index_i,
new_index_j)

# Display the result
print(f"Dot product of the {new_index_i}-th and {new_index_j}-th
eigenvectors: {result_new}")
```

```
Dot product of the 0-th and 1-th eigenvectors: -0.293689239285741
```

1)c)

```
#1)c)
import numpy as np

def my_transf(A):
    return A + np.eye(A.shape[0])

def eigprod(A, i, j):
    eigenvalues, eigenvectors = np.linalg.eig(A)
    return np.dot(eigenvectors[:, i], eigenvectors[:, j])

np.random.seed(20)
A = np.random.randn(4, 4)
```

```
print("Random Matrix A:")
print(A)

i = 1
j = 0
result1 = eigprod(my_transf(A), i, j)
print(f"\nDot product with i={i}, j={j}: {result1}")
```

```
Random Matrix A:
[[ 0.88389311  0.19586502  0.35753652 -2.34326191]
 [-1.08483259  0.55969629  0.93946935 -0.97848104]
 [ 0.50309684  0.40641447  0.32346101 -0.49341088]
 [-0.79201679 -0.84236793 -1.27950266  0.24571517]]

Dot product with i=1, j=0: -0.7356383218467829
```

1)d)

Let's analyze the given expression $\left(v_2(R)^T \cdot (R^T - R) \cdot v_3(R)\right)$.

Given the context of eigenvalues and eigenvectors, let's break down the reasons for this expression:

1. **Eigenvector Orthogonality:**
   - Eigenvectors corresponding to distinct eigenvalues of a real symmetric matrix are orthogonal to each other.
   - Let $(v_2(R))$ and $(v_3(R))$ be the eigenvectors corresponding to distinct eigenvalues $(\lambda_2(R))$ and $(\lambda_3(R))$, respectively.
   - The transpose of $(v_2(R))$ is denoted as $(v_2(R)^T)$.
   - Since $(\lambda_2(R))$ and $(\lambda_3(R))$ are distinct, $(v_2(R))$ and $(v_3(R))$ are orthogonal.
2. **Properties of Symmetric Matrices:**
   - If (R) is a real symmetric matrix, then $(R^T = R)$.
   - Therefore, $((R^T - R))$ is a zero matrix.

Now, let's analyze the expression:

$$\left[v_2(R)^T \cdot (R^T - R) \cdot v_3(R)\right)$$

- Due to the orthogonality of $(v_2(R))$ and $(v_3(R))$, the product $(v_2(R)^T \cdot v_3(R))$ is zero, as the dot product of orthogonal vectors is zero.

- Additionally, since $((R^T - R))$ is a zero matrix for symmetric matrices, the entire expression evaluates to zero.

Therefore, the key observation here is that the given expression is zero, which can be attributed to the orthogonality of eigenvectors corresponding to distinct eigenvalues and the properties of symmetric matrices.

1)e)

To explore whether the eigenvectors of $(\text{my\_transf}(A))$ are linearly dependent, let's denote the eigenvectors of $(\text{my\_transf}(A))$ as $(v_1, v_2, \dots, v_n)$, corresponding to the eigenvalues $(\lambda_1, \lambda_2, \dots, \lambda_n)$.

Given that $(\text{my\_transf}(A) = A + I)$, where (I) is the identity matrix, the eigenvalues $(\lambda_i(\text{my\_transf}(A)))$ are simply $(\lambda_i(A) + 1)$ because adding the identity matrix to (A) shifts all eigenvalues by 1.

Now, let $(v_i(\text{my\_transf}(A)))$ be the eigenvector of $(\text{my\_transf}(A))$ corresponding to $(\lambda_i(\text{my\_transf}(A)))$.

If $(v_1(\text{my\_transf}(A)))$ is linearly dependent on $(\{v_2(\text{my\_transf}(A)), \dots, v_n(\text{my\_transf}(A))\})$, it means we can express $(v_1(\text{my\_transf}(A)))$ as a weighted sum of $(\{v_2(\text{my\_transf}(A)), \dots, v_n(\text{my\_transf}(A))\})$, i.e.,

$$\left[v_1(\text{my\_transf}(A)) = \sum_{k=2}^{n} a_k \cdot v_k(\text{my\_transf}(A))\right)$$

Now, let's take the dot product of both sides with $(v_j(\text{my\_transf}(A)))$ ¿ is any index from 2 to (n) ¿ :

$$\left[\langle v_1(\text{my\_transf}(A)), v_j(\text{my\_transf}(A))\rangle = \sum_{k=2}^{n} a_k \cdot \langle v_k(\text{my\_transf}(A)), v_j(\text{my\_transf}(A))\rangle\right)$$

The left side is $(\delta_{1j} \cdot \| v_1(\text{my\_transf}(A)) \|^2)$, and the right side involves dot products of eigenvectors with distinct indices. If $(v_1(\text{my\_transf}(A)))$ is linearly dependent on the other eigenvectors, this dot product will not be zero for any (j). Otherwise, if $(v_1(\text{my\_transf}(A)))$ is linearly independent, this dot product will be zero for $(j \neq 1)$.

1)f)

- If all eigenvectors are linearly independent, the dimension of the vector space is equal to the total number of eigenvectors.
- If at least one eigenvector is linearly dependent, the dimension of the vector space is less than the total number of eigenvectors.

1)g)

To determine whether the given three vectors form a basis for $\left(R^3\right)$, we need to check if they are linearly independent. The vectors are:

$$\left[v_1=\left[\cos^2(\theta),2,-1\right]\right)$$

$$\left[v_2=\left[\sin^2(\theta),-2,1\right]\right)$$

$$\left[v_3=\left[1,6,-3\right]\right)$$

The vectors form a basis for $\left(R^3\right)$ if and only if they are linearly independent.

To check for linear independence, we can set up the following linear combination:

$$\left[c_1 v_1+c_2 v_2+c_3 v_3=0\right)$$

where $\left(c_1,c_2,c_3\right)$ are constants, and (0) is the zero vector in $\left(R^3\right)$.

Substitute the vectors and set up the system of equations:

$$\left[c_1 \cos^2(\theta)+c_2 \sin^2(\theta)+c_3=0\right)$$

$$\left[2c_1-2c_2+6c_3=0\right)$$

$$\left[-c_1+c_2-3c_3=0\right)$$

This system of equations represents the conditions for linear independence. The vectors form a basis if the only solution to this system is $\left(c_1=c_2=c_3=0\right)$.

Now, by analyzing the system of equations and consider the values of (θ) for which the only solution is the trivial solution.

Since the determinant is 0, any value of θ won't satisfy the matrix.

Question 2

2)a)

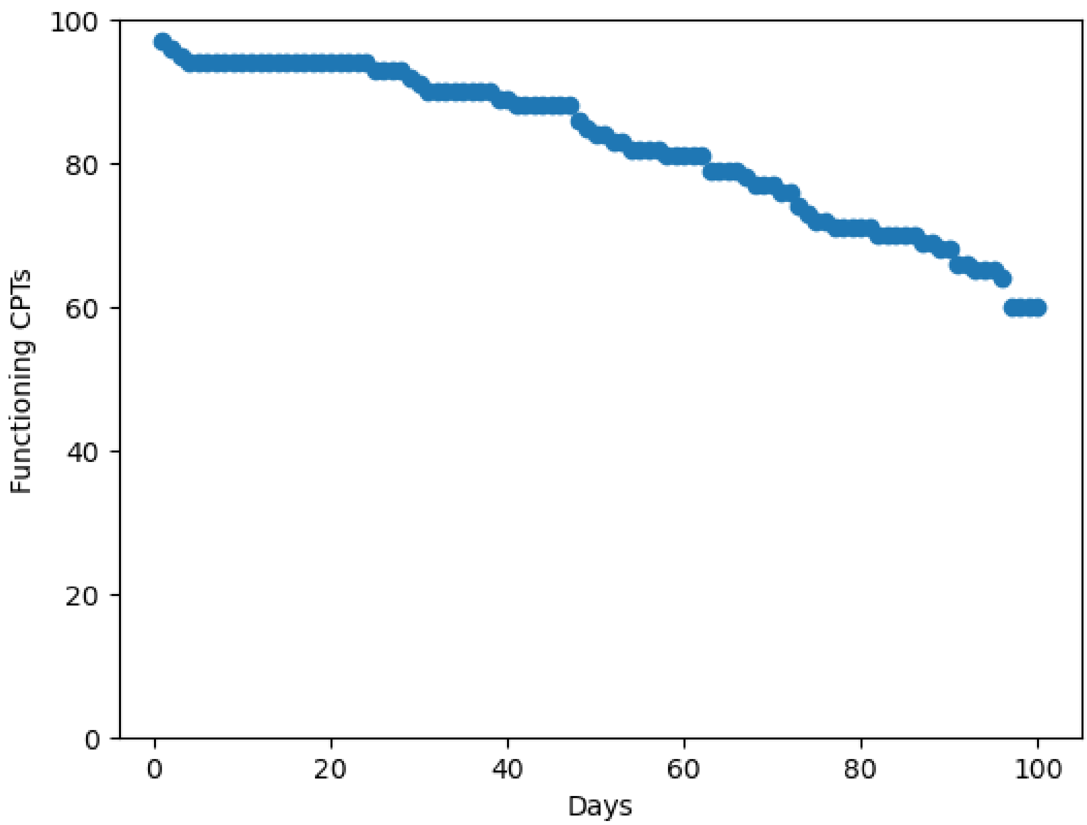
```
#2)a)
from math import prod

def calculate_remain_probability():
    probability_overuse = [0.0001 * t**2 / (1 + t) for t in range(0,
100)]
    probability_factory = [0.01 * (1 + (1 - t) / (1 + t)) for t in
```

```
range(0, 100)]
    combined_probabilities = probability_overuse + probability_factory
    survival_probability = prod(1 - (np.array(probability_overuse) +
np.array(probability_factory)))
    return survival_probability

result_changed_variables = calculate_remain_probability()
print(result_changed_variables)
```

```
0.5535619154108106
```

2)b)

```
#2)b)

def simulate_factory(n_initial):
    functioning_cpts = np.zeros(100, dtype=int)
    n = n_initial
    cpt_count = []

    probability_overuse = [0.0001 * t**2 / (1 + t) for t in range(0,
100)]
    probability_factory = [0.01 * (1 + (1 - t) / (1 + t)) for t in
range(0, 100)]

    for day in range(100):
        probability_failure = probability_overuse[day] +
probability_factory[day]
        n = np.random.binomial(n, 1 - probability_failure)
        cpt_count.append(n)

    return cpt_count

cpts_count_result = simulate_factory(100)
plt.scatter(np.arange(1, 101), cpts_count_result)
plt.xlabel("Days")
plt.ylabel("Functioning CPTs")
plt.ylim(0, 100)
plt.show()
```



2)c)

```
#2)c)
def profit(x, y):
    # Given probabilities of overuse and factory faults
    p_overuse = [0.0001 * t**2/(1+t) for t in range(1,101)]
    p_factory = [0.01* (1+(1-t)/(1+t)) for t in range(1,101)]

    # Number of items
    n_items = 100

    # Initialize total expected faults
    expected_factory_faults = 0
    expected_overuse_faults = 0

    # Calculate the expected number of faulty items for each month
    for t in range(100):
        p_factory_fault = p_factory[t] * x
        p_overuse_fault = p_overuse[t] * y
        expected_factory_faults += n_items * p_factory_fault
        expected_overuse_faults += n_items * p_overuse_fault

    # Calculate the total expected profit
```

```
    expected_profit = n_items * 5 - (expected_factory_faults + 
expected_overuse_faults) * 5

    return expected_profit

# Example usage:
investment_in_factory = 0.3  # Replace with the actual investment 
value for reducing factory faults
investment_in_overuse = 0.7  # Replace with the actual investment 
value for reducing overuse faults

result = profit(investment_in_factory, investment_in_overuse)
print(f"The expected profit for 100 items is: £{result:.2f}")

The expected profit for 100 items is: £314.01
```

2)d)

Let's say:

- (x) as the investment per CPT in reducing overuse faults,
- (y) as the investment per CPT in reducing factory faults.

The profit function (p(x, y)) represents the net profit from each CPT after considering the investments in fault reduction.

The optimization problem can be formulated as follows:

$\left[\text{Maximize } p(x, y) \text{ subject to } 0 \leq x \leq 0.10 \text{ and } 0 \leq y \leq 0.10\right)$

This means the company wants to maximize the profit function while adhering to the constraint that the total investment per CPT in fault reduction cannot exceed 10 pence.

For a locally optimal allocation of investment, the condition on the gradient (or Jacobian) of (p) should be that it is equal to the zero vector. In mathematical terms, at a locally optimal solution $\left(\left(x^{¿}, y^{¿}\right)\right)$, the gradient (or Jacobian) of (p) should satisfy:

\begin{align*} |frac{dp}{dx} &= 0 | |frac{dp}{dy} &= 0 |end{align*}

This condition represents a critical point where the partial derivatives of (p) with respect to (x) and (y) are zero, indicating a potential maximum or minimum. Checking the second-order conditions (e.g., the Hessian matrix) can further confirm whether it is a maximum or minimum.

Question 3

Game1:

3)a)

The outcome space, or sample space, for this scenario involves two uncertain events:

1. The selection of the box that contains the money (BR). There are three possible outcomes for this event:

BR = B1, BR = B2, or BR = B3.

1. The choice of the box made by the player (Bc). Once the box with the money is determined, the player can choose any of the three boxes. Therefore, there are three possible outcomes for this event:

Bc = B1, Bc = B2, or Bc = B3.

The sample space can be represented as the set of all possible pairs (BR, Bc): {(B1, B1), (B1, B2), (B1, B3), (B2, B1), (B2, B2), (B2, B3), (B3, B1), (B3, B2), (B3, B3)}.

3)b)

To calculate the expected reward (E[R]) and the variance (Var(R)), we need to consider all possible outcomes and their associated rewards.

Let (X) represent the amount of money in the chosen box (BR), and let (R) represent the reward. The possible outcomes and rewards are as follows:

- If $(Bc = BR)$, then (R = X) (you earn the amount in the chosen box).
- If $(Bc \neq BR)$, then (R = 0) (you earn nothing).

Now, let's calculate (E[R]) and (Var(R)):

$$\left[E[R] = \sum_i P(\text{Outcome } i) \cdot R_i\right)$$

$$\left[Var(R) = \sum_i P(\text{Outcome}\, i) \cdot (R_i - E[R])^2\right)$$

Given that each box is chosen with uniform probability, $\left(P(\text{Bc = BR}) = \frac{1}{3}\right)$ and \$(P(\text{Bc \neq BR}) = \frac{2}{3})\$.

Substitute these probabilities into the formulas and perform the calculations. The result will provide the expected reward (E[R]) and the variance (Var(R)).

Game 2:

3)c)

Let's analyze the 'stick' and 'swap' strategies separately.

Stick Strategy:

- If you initially choose the correct box (BR = Bc), you will stick with it, and your reward will be the amount of money in the chosen box (R = X).
- If you initially choose the incorrect box (BR ≠ Bc), opening one empty box doesn't change your choice, and your reward remains zero (R = 0).

Let $(P(\text{Initial Correct}))$ be the probability that you initially choose the correct box and $(P(\text{Initial Incorrect}))$ be the probability that you initially choose an incorrect box.

$$[E[R \vee \text{Stick}] = P(\text{Initial Correct}) \cdot X + P(\text{Initial Incorrect}) \cdot 0]$$

$$[Var(R \vee \text{Stick}) = P(\text{Initial Correct}) \cdot (X - E[R \vee \text{Stick}])^2 + P(\text{Initial Incorrect}) \cdot (0 - E[R \vee \text{Stick}])^2]$$

Swap Strategy:

- If you initially choose the correct box (BR = Bc), opening one empty box doesn't change your choice, and your reward remains zero (R = 0).
- If you initially choose an incorrect box (BR ≠ Bc), the opened empty box reveals the location of the correct box. When you swap, your reward becomes the amount of money in the unchosen box (R = X).

$$[E[R \vee \text{Swap}] = P(\text{Initial Correct}) \cdot 0 + P(\text{Initial Incorrect}) \cdot X]$$

$$[Var(R \vee \text{Swap}) = P(\text{Initial Correct}) \cdot (0 - E[R \vee \text{Swap}])^2 + P(\text{Initial Incorrect}) \cdot (X - E[R \vee \text{Swap}])^2]$$

3)d)

For the 'swap' strategy with K boxes containing a reward out of N total boxes:

$$\left[E[R \vee \text{Swap}] = P(\text{Initial Incorrect}) \cdot \frac{X}{N-1}\right]$$

$$\left[Var(R \vee \text{Swap}) = P(\text{Initial Incorrect}) \cdot \frac{X^2(N-K)}{(N-1)^2}\right]$$

These formulas consider the probability of initially choosing an incorrect box and then swapping to find the reward.

Let's take at look in detail:

- The probability of initially choosing a box with a reward is $\left(P(X=1)=\frac{K}{N}\right)$, while the probability of initially choosing a box without a reward is $\left(P(X=0)=\frac{N-K}{N}\right)$.
- If you initially choose a box without a reward and then swap, the probability of getting a reward after swapping is $\left(P(X=1 \text{or swap})=\frac{K-1}{N-1}\right)$.
- If you still choose and swap the box without a reward, the reward probability is $(P(X=1 \text{or swap})=1)$.
- However, if you initially choose a box with a reward and then swap, you will always end up with an empty box, resulting in a probability of 0.
- The overall probability of getting a reward after swapping doesn't change; it remains $\left(P(X=1)=\frac{K}{N}\right)$.
- The conditional probability of getting a reward after swapping depends on whether you initially choose a box with a reward or not.
- If you swap after choosing a box with a reward, you will always receive an empty box.
- The expected value is given by $\left(P(X=1)\cdot 1+P(X=0)\cdot P(X=1 \text{ or swap})=\frac{16}{54}\right)$.
- The variance is given by $\left(P(X=1)\cdot(1-\text{expected value})^2+P(X=0)\cdot(P(X=1 \text{or swap})-\text{expected value})^2=\frac{1}{54}\right)$.

Question 4

4)a)

The maximum predator population for which prey births still outstrip deaths can be found by setting the prey birth rate equal to the prey death rate:

[aN - bNP = 0]

Solving for (N), we get $\left(N=\frac{b}{a}\right)$. Therefore, the maximum predator population is $\left(\frac{b}{a}\right)$.

4)b)

The predator population is shrinking when the predator death rate exceeds the predator birth rate. This occurs when $(d > c\,N)$, meaning that the rate at which predators die is greater than the rate at which they are born.

4)c)

- In the absence of predators (P(0) = 0), the dynamics of the prey population over time would be governed solely by the prey birth and death rates.
- The prey population would grow exponentially in the absence of predation pressure.
- This is reasonable as, without predators, there is no factor limiting the growth of the prey population.

4)d)

For the modified predator-free prey dynamics:

$$\left[N(t) = a\,N(t)\left(1 - \frac{N(t)}{K}\right)\right)$$

The fixed points occur when the prey birth rate equals the prey death rate:

$$\left[a\,N\left(1 - \frac{N}{K}\right) = 0\right)$$

This equation has two fixed points: $(N=0)\,and\,(N=K)$. The parameter (K) represents the carrying capacity of the environment for the prey population. It is the maximum population size that the environment can sustain in the absence of predators. The term $\left(1 - \frac{N}{K}\right)$ represents the factor limiting the prey population's growth as it approaches the carrying capacity (K).

```
data =
np.array([0.0,0.8253705906272355,0.8202189457560609,0.01,1.25125933369
67723,0.8804750446630152,0.02,1.0744696556474873,1.1951921572383182,0.
03,1.1459487035702027,1.367022776863449,0.04,0.7406895127926447,0.8201
930700228012,0.05,0.8032848443220044,0.8020943063852652,0.06,1.0423448
720147894,0.9964271033305258,0.07,0.8669728720976302,0.944155919286786
3,0.08,1.2485960741022775,1.5041279662393712,0.09,1.076411156475735,1.
1868935239104308,0.1,1.0913514213811686,0.9337625654623029,0.11,1.2211
70545353793,1.1610562276254386,0.12,1.375609475040752,0.65599129204274
3,0.13,0.8456932534642474,0.8506565612423291,0.14,1.258988003318481,1.
143765935311O625,0.15,1.0879326273705237,1.0144919079948402,0.16,1.215
3693242032464,0.7890236302137076,0.17,1.3345684906603732,0.84542497857
```

5131,0.18,1.3137461562592339,0.8732260333351302,0.19,1.428277636494295
,0.7061086355004339,0.2,1.5634075717811937,1.0395408458276172,0.21,1.3
657438416115804,0.7780988598914812,0.22,1.4701608691245298,0.543739224
0834374,0.23,1.5807693768769069,0.9623069207018562,0.24,1.361188334734
7324,0.9821342334236914,0.25,1.445754305632874 8,0.846371016266850 1,0.2
6,1.513272170291705,0.45113076913045824,0.27,1.5756694495348156,0.7030
469767603075,0.28,1.7020605367742399,0.5729183278716372,0.29,1.5481502
977674486,1.0163316238356903,0.3,1.5900510345416954,1.1060720781878284
,0.31,1.0176651770385163,0.8433681203723178,0.32,1.4750305241179156,0.
8633310778531771,0.33,1.7081457157589555,0.7183105081906727,0.34,1.454
5873580628512,0.7161981206236734,0.35,1.6425374522443608,0.84516697129
66948,0.36,1.2322095928749066,0.7277924775899506,0.37,1.74422368026962
58,0.9444102056298844,0.38,1.6192842786863628,0.521116022711692,0.39,1
.6720580362341673,0.9848395057495227,0.4,1.4406205509598402,0.65156166
76522077,0.41,1.8104217696306768,0.8706134751492154,0.42,1.74599320315
71956,0.3144483218946911,0.43,1.5767615322608257,0.9555716460005269,0.
44,1.9883647007071132,0.6516479213791484,0.45,2.0759946358397485,0.358
7906462339842,0.46,1.7891540869010643,0.7250806159755979,0.47,2.006394
0831177627,0.6774129096864877,0.48,2.351207088675081,1.023650946059681
4,0.49,2.0023608091194363,0.667932869715822,0.5,1.7307173612759736,0.5
01168623020454,0.51,2.039375219294437,0.8649538899170633,0.52,1.794070
3111564535,0.7907339894164709,0.53,1.8975440140190953,1.03970298149104
77,0.54,2.1292376751093736,0.5619762412364976,0.55,2.166703207978072,1
.0916421814821662,0.56,2.355302913074924,0.6074458043963602,0.57,2.508
1045622410763,0.8992398806684452,0.58,2.4655037772173114,0.71001321471
71605,0.59,2.3719926113777676,0.9543949865179597,0.6,2.415344505684415
,0.7608814073283383,0.61,2.4519116927962483,0.1518977986382677,0.62,2.
508686203478478,0.8777536036963025,0.63,2.44182495142244,0.38237907161
56763,0.64,2.0953200090463766,0.756025640199692,0.65,2.371777573831583
,0.94151456106178,0.66,2.7044765777810214,0.26000283454630163,0.67,2.7
806801024780663,0.4655487715362591,0.68,2.8505994601079347,0.873101566
1504052,0.69,2.118623542208117,0.8223683742763328,0.7,2.62165042886252
9,0.9131445555351273,0.71,2.582402132244429,0.8946045465353925,0.72,2.
483455037038926,1.0804949294453423,0.73,2.7572845455280195,0.878101892
2046983,0.74,3.037439215513899,0.7605951160308121,0.75,2.8142606623509
803,0.5718944238558733,0.76,2.882627355854355,0.8423085744922739,0.77,
2.6919362643369737,0.6537816111465011,0.78,2.9753950748656632,0.719193
181569344,0.79,3.0201614018936467,0.9311606867703295,0.8,2.83785951797
97887,1.3062887919414568,0.81,3.1485048063642074,0.9583632891106408,0.
82,2.8551434774933604,0.5686122074721297,0.83,2.7217156460548937,0.791
6533287783812,0.84,3.0066794823419003,0.7229932623718125,0.85,3.299192
9070971246,0.7708487675653453,0.86,3.2554299528848207,0.88261917399387
39,0.87,3.350693449957946,0.6764175410308613,0.88,3.1485729050577214,1
.0557116508974644,0.89,2.940802542097215,1.3186954675236489,0.9,3.4216
160463483165,1.0655201490268742,0.91,3.6496661453401464,0.986828858098
5981,0.92,3.7735996493671515,0.6236263561003901,0.93,3.677925314438037
5,1.0522258249795788,0.94,3.5548241659620192,0.9970048790818771,0.95,3
.7205692095695384,0.9363674379234116,0.96,3.7191478156316724,1.1807101
54914114,0.97,3.8265182016340824,1.0614030299339015,0.98,3.76541749657

18243,1.0809638766740044,0.99,3.8876378228393293,0.8805970655863106,1.
0,3.2517154072100416,1.061573048050119,1.01,3.802494373562257,1.294143
355228126,1.02,3.7380941484585244,1.4174672336551806,1.03,4.1041909494
9246,1.2897930568291085,1.04,3.8610475178369366,1.3813135272991088,1.0
5,3.90655891410061,0.9567145819063867,1.06,4.116352164311322,0.9255520
53189032,1.07,3.785241723237567,1.2679570691628406,1.08,3.738457683204
6372,1.123604703705015,1.09,4.0399392139651935,1.4784111890087166,1.1,
3.9920674027997056,1.280541928801772,1.11,3.761001739376474,1.29989155
51104322,1.12,4.179086933311257,1.3027816505973397,1.13,3.693889755996
9477,1.0357324781825141,1.14,3.865997460139013,1.486026073023141,1.15,
4.126991087506379,1.3366818924274007,1.16,4.44028835829517,1.302071847
268636,1.17,3.8948317213944947,1.2719272880533414,1.18,3.9660739831477
807,1.5269101959997764,1.19,3.96417055783056,1.394160532189975,1.2,3.9
51741379634038,1.3393730340995145,1.21,4.130423700118754,1.92252940775
84258,1.22,4.011153295523203,1.3789558003503761,1.23,4.127114937276425
,1.8158897289687976,1.24,3.8576560383714424,1.7175629765004268,1.25,4.
028234337453515,1.5799859944613261,1.26,4.245641116516499,2.0049858007
88266,1.27,4.076117409545003,1.9105867315709613,1.28,4.00400301175127,
1.6556424770499498,1.29,3.7823342308538295,1.6324364627833166,1.3,3.76
4996086913561,2.1247393370528167,1.31,3.8659291830929945,1.87843508332
46848,1.32,3.5968682859804626,2.188643095611304,1.33,3.903770860376655
,2.111913062251637,1.34,3.507049053282115,2.2999758180173573,1.35,3.60
2888291346972,1.9809206761801883,1.36,3.2503668252016253,2.17937602726
6853,1.37,3.2415726746318536,1.8744057617442926,1.38,3.298615446389396
,2.2443323580979375,1.39,3.3496050279320624,2.746262416501709,1.4,3.55
23062176565197,2.375882352440953,1.41,3.356074635593233,2.473199889392
5824,1.42,3.124247954671158,2.109360093874913,1.43,2.8411526022537363,
2.690627874902785,1.44,3.051096033186018,2.492938663675386,1.45,3.0762
4609786167,2.108214342756693,1.46,2.9047865239434705,2.417236745149951
,1.47,3.0559491404706045,2.46153937847029,1.48,2.75546186847479,2.6386
13254380666,1.49,2.965551628445072,2.5230261567288754,1.5,2.6184637344
935116,2.840825402995792,1.51,2.4865336280057253,2.5942899295575756,1.
52,2.863511151956073,2.7074844267779614,1.53,2.471359518548605,2.41456
2391126137,1.54,2.3413930374187584,2.520526907526854,1.55,2.5956239512
418406,2.58576815527428,1.56,1.899129933067544,2.6652152687682444,1.57
,2.736308054048812,2.6743762721684883,1.58,2.1619347853076345,2.630687
460093294,1.59,2.3017624464496675,2.6023090062623413,1.6,2.16683324032
87187,2.6066345070118317,1.61,2.2313821910680023,2.897725782442959,1.6
2,2.1627966876233238,2.7052575832788057,1.63,2.291740372742384,2.70437
21875560878,1.64,2.2921606630342897,2.912388404601338,1.65,1.888792181
2254713,2.892475740447905,1.66,1.646931925264305,2.7376866240869235,1.
67,1.4503187834012772,2.881901979381492,1.68,1.5897447161567788,2.5837
59618163891,1.69,1.7731465796823263,3.0635672260641806,1.7,1.652301673
8776406,2.8226253701502704,1.71,1.1990886777888394,3.04477240864657,1.
72,1.3029898833844376,2.7631376192260317,1.73,1.4537400757641468,2.398
5093131095767,1.74,1.889872090711736,2.4667584870661416,1.75,1.1574287
858045684,2.583915356682998,1.76,1.6432197897921526,2.4644503111419134
,1.77,1.516895712695242,2.7214622407673947,1.78,1.4640042795400443,2.4
04350028043953,1.79,1.1849935631094908,2.4395917381343404,1.8,1.175119

```
5534749899,2.4442755167909045,1.81,1.2550483535239274,2.38071147389449
23,1.82,1.3285483741775175,2.525899488020486,1.83,0.9065771882406435,2
.276068167668241,1.84,1.3721556067161755,2.908165170703417,1.85,1.4019
46604385084,2.212138315037437,1.86,1.332569785408628,2.588678931149495
,1.87,0.9665648235610689,2.6303059297285527,1.88,1.45424514734988,2.58
31852858705266,1.89,0.8508406820771799,2.419095330210788,1.9,0.9782665
372929478,2.421449454871508,1.91,1.2271608150569089,2.1249827276443902
,1.92,1.3009739216215368,2.1044471944866685,1.93,1.3878758883141558,2.
274632068316371,1.94,1.0192196455894986,2.3631181318447405,1.95,0.7460
924551769945,2.212994962159642,1.96,1.0135600493536732,2.0991988740301
75,1.97,1.357768301252331,1.8143388500069082,1.98,0.8458840313979914,2
.2490630949281045,1.99,0.5837888734093417,1.798069126839448,2.0,1.1662
91175149156,1.752280256424795,2.01,1.2639980921120724,2.22383149975478
9,2.02,0.7439375366871275,1.9681020078763587,2.03,1.0322946391382744,2
.2284417182561764,2.04,1.1135931690994436,2.3167235296122604,2.05,0.70
63244789844492,2.15162876070362,2.06,1.028004626103985,1.7400111941800
125,2.07,0.7643890567290035,2.247327313264645,2.08,0.6119526070979482,
2.0563119892544495,2.09,0.8206242707710532,1.9357572375683454,2.1,0.81
91755713776503,1.863006116121921,2.11,0.8155354463512944,2.38698686074
71116,2.12,1.0103225502097959,1.9695111087867316,2.13,1.29757289374864
68,2.081289749140055,2.14,0.7798735514758816,2.1153312383253966,2.15,0
.7189005245823479,1.6465468387977473,2.16,0.6986417935362176,1.6851321
022741892,2.17,0.690833148546571,1.8496259015168497,2.18,1.06276772511
38892,1.7670672162246546,2.19,0.6512178027905718,1.8668013001146722,2.
2,0.6509854672497042,2.001288365242748,2.21,0.5677029315528865,1.57504
74195633095,2.22,0.3120192685526839,1.6158642662108906,2.23,1.13893174
45455605,1.6540484996323912,2.24,0.7243157673843392,1.8454300487432365
,2.25,0.5299175819196007,1.4093341691005568,2.26,0.8859130924104746,1.
4946593886760249,2.27,1.0742787196198258,1.7262096340820579,2.28,0.923
7842656511487,1.386343279096365,2.29,1.0627867649960023,1.161568974900
8193,2.3,0.7194877974007753,1.9406547702604644,2.31,0.6474155677173562
,1.3435399470045093,2.32,0.583547630831506,1.1381175796749943,2.33,0.5
325677412539862,1.730938166142046,2.34,1.0085575774117688,1.6202944332
089906,2.35,0.887860657708986,1.4395796076161362,2.36,1.17315471360750
07,1.2704364915263435,2.37,0.46333070178271596,1.333505285789843,2.38,
0.9281905641170881,1.473859255307826,2.39,1.0953940529806867,1.7779420
80387186,2.4,0.7120634357997596,1.3367259818483956,2.41,0.782530853082
0887,1.7215577322244107,2.42,0.7409897423784212,1.310485182902497,2.43
,0.5858010007075899,1.186145201514079,2.44,1.1244774914677729,1.184862
9753421618,2.45,0.6942398731358219,1.2858357708633905,2.46,0.749159984
0603169,1.4038381422528943,2.47,0.6289297315813736,1.4116877229509261,
2.48,1.0396097922561618,1.050591374030939,2.49,0.7533200282638459,1.48
09033431435237,2.5,0.7622777858502037,0.7808348824142383,2.51,1.100116
9171522376,1.3218779124026583,2.52,1.3708768106059916,1.14100902916026
28,2.53,1.0729989830951117,1.1917579450667932,2.54,0.7635988914072376,
1.16425282984012,2.55,1.137816537261179,1.4443939617294597,2.56,0.9065
717790346248,1.04844333378782,2.57,0.7097657118429275,1.17625081847049
14,2.58,0.7426919242906024,1.0527987825612088,2.59,0.8919704126214323,
1.1514113180329035,2.6,0.5203822104718214,1.0231602470759045,2.61,0.75
```

```
84046867588214,1.1857603134058605,2.62,0.7117511373934757,0.8942012697
446813,2.63,0.8139187043410363,0.835381263061259,2.64,0.93058549857853
32,1.0410799865569764,2.65,0.7720714250170179,0.7399813061387854,2.66,
0.8665655337958041,0.9184115189473784,2.67,0.6226243356836769,0.966742
8541531915,2.68,0.8942032431208679,1.2674375245039973,2.69,1.070955802
0956051,1.3577172095867671,2.7,0.914859042690237,1.0489949912513266,2.
71,1.383860415035373,0.7557940062672551,2.72,1.0792281257663257,1.0146
44374566306,2.73,0.7927760626973879,0.7314435322425165,2.74,1.06388407
6395786,0.5418285971326535,2.75,1.0626614775268373,0.48332479623047087
,2.76,0.6957754676278398,0.8386053181770811,2.77,0.7336209352788814,0.
7231693671889614,2.78,1.020578250695643,0.6432999286751863,2.79,0.9863
716780108929,0.8165026965676865,2.8,1.1449456542864722,1.1348251902471
35,2.81,1.1801440560533858,0.9666140623127432,2.82,1.2566603583572178,
0.7387823222754977,2.83,1.2530430246308948,0.9256490220874591,2.84,1.3
64016710639729,0.9573524700156668,2.85,1.3023833732259464,1.1388448153
319857,2.86,1.101350481362554,0.7539036301948927,2.87,0.73071000106419
53,0.8553560568141589,2.88,1.2355721675143472,0.8222929387654909,2.89,
1.1887587906213644,0.7315498755268175,2.9,1.220400666008882,0.67025425
2029709,2.91,1.5400681158220777,0.799592336899042,2.92,1.3822247150671
003,0.8228597376661854,2.93,1.280601782915936,0.9390809570568438,2.94,
1.2662338818566479,0.9517682733305673,2.95,0.9101946917356465,0.842925
326536793,2.96,1.2869340840183179,0.6866093918525702,2.97,1.7834865660
352501,0.8580123877142334,2.98,1.6054485098373064,0.4561651612546335,2
.99,1.4553563777136496,0.8730845983491127,3.0,1.3886186163865049,0.937
0772110223358,3.01,1.5692575550285557,0.7639996777821291,3.02,1.137510
067505801,0.7720867949226508,3.03,1.3795294210196107,0.490598384237143
25,3.04,1.500497546361021,0.3387571812392564,3.05,1.7443111543940788,0
.6015884213906206,3.06,1.500498228519984,0.5034183917045102,3.07,1.656
395319783736,0.47708158515290017,3.08,1.623932221995836,0.504297000639
611,3.09,1.303805875916974,0.5181466590241512,3.1,1.4449304052094298,0
.8270741717504877,3.11,1.6928401043591466,0.3901195933890724,3.12,1.65
09962530875657,0.7180548677903247,3.13,1.711360621939023,0.85381643162
64223,3.14,1.7897762872010488,0.7561677019783622,3.15,1.86268645906976
58,0.8644633585047476,3.16,2.2409192010909718,0.49020707536618013,3.17
,2.178737498458009,0.5823152290590482,3.18,2.2273677811923163,0.685862
2961862559,3.19,1.9219700742112877,0.5858021433830191,3.2,2.0443163794
844326,0.5460773348879607,3.21,1.997734129180287,0.84890636842309,3.22
,1.629770275039083,1.0496813686582676,3.23,1.9210844279677886,0.787995
2247575518,3.24,1.8270647239115714,0.4485162980542294,3.25,2.002861761
7485027,0.39309160583724717,3.26,2.0255790374929297,0.8734757506062962
,3.27,2.048743315405754,0.6341073542659182,3.28,2.3296649475413904,1.0
17477004166842,3.29,2.7317447929930134,0.8569936126503923,3.3,2.354544
0709081893,0.7941761502836587,3.31,2.2200099626199785,0.92904814726812
27,3.32,2.236920329140511,0.631237093447086,3.33,2.3014687000830265,0.
7601073638463262,3.34,2.6686529380394832,0.9864570457142519,3.35,2.639
314615105396,0.5138140374059968,3.36,2.3909683874272494,0.884314364399
9684,3.37,2.795671934445007,1.006561115596391,3.38,2.8075247468439346,
0.5584016868092817,3.39,2.5857492264308672,0.8271407317951943,3.4,2.71
22844150811813,0.831107983496409,3.41,2.7758981858274288,0.63682893591
```

```
24656,3.42,2.8336495712909953,0.9760641683501623,3.43,2.83293325115350
92,1.033342226447769,3.44,3.2432071696829228,0.8259589450474693,3.45,2
.773018178763293,0.7611846982504704,3.46,2.8044881246711184,0.88182342
25544309,3.47,2.956566402450238 5,0.8260104863251989,3.48,2.99143537206
267,0.6791430721277019,3.49,3.0387032610249864,0.6599077046917755,3.5,
2.9080108143605834,0.8640889732846994,3.51,3.1716913376795715,0.816622
9731247442,3.52,3.5677310613244617,0.7513862244503224,3.53,3.383604932
803528,0.6878886161628661,3.54,3.4441828863193718,0.7755396089443504,3
.55,3.444555090187697,0.9380211121671975,3.56,3.2401656735791686,1.083
9269382599375,3.57,3.439796769572349,1.207469724065135,3.58,3.31884994
5834521,0.9304072651777695,3.59,3.2995786165072696,0.4942899113165482,
3.6,3.430586194791578,0.6498288812686284,3.61,3.406492772358093,0.9154
851168347139,3.62,3.7118881941370545,1.1265674376950512,3.63,3.5717604
23485351,1.058732912571716,3.64,4.155657472725945,0.9535227996806671,3
.65,3.8144563747700464,0.8681669914953192,3.66,3.9359492051017284,1.21
69910128359345,3.67,3.528590720958142,1.1499030019508165,3.68,4.109905
553120379,1.0638960644088797,3.69,4.045796334545698,1.2758236404292198
,3.7,3.6724529253102602,0.8883403654819115,3.71,4.046005255086986,1.05
41405541350555,3.72,3.7765416726333374,1.2498624853768905,3.73,4.46997
1652862149,1.0839307566674579,3.74,4.228275437245985,1.406502590760931
9,3.75,3.9488106848863045,1.2206619281141573,3.76,4.0482166435395825,1
.6760275041191273,3.77,4.296878335291239,1.195774276825841,3.78,4.4051
13543184753,1.1021281875174112,3.79,3.729844395047192,1.61614458843888
54,3.8,4.496489870419789,1.3164685359394497,3.81,4.577045214012909,1.5
937349352866683,3.82,4.2154883912117835,0.891815430538095,3.83,4.17247
5827179526,1.5625910908347291,3.84,4.3234605310432395,1.47596253003800
06,3.85,3.7332067626883596,1.5284533326681207,3.86,4.125194631042692,1
.472865908342905,3.87,4.880353338049279,1.155676809943731,3.88,4.36838
28339914506,1.6968577168851948,3.89,4.2759461530536305,1.6974507758126
047,3.9,4.281495835898362,1.80238117940915,3.91,4.69074591641653,1.485
7277164157008,3.92,4.407367613585688,1.7525501116517788,3.93,3.9881134
307669868,1.9517704498860997,3.94,4.15938001496773,1.6971603655143466,
3.95,4.216406285045702,1.7065194432954738,3.96,4.3123938601945415,1.98
78665766349222,3.97,3.9713570858192067,1.6396698990704897,3.98,4.32491
7713617724,2.065545224343898,3.99,4.398278780768667,1.9932667056741626
,4.0,3.714355957000438,2.1669482725150457,4.01,3.6541713623597656,1.95
1367676789600 9,4.02,3.974955729964032,2.0427383751060884,4.03,4.265143
785731212,2.232554724994495,4.04,3.8528193970591813,1.9568140095455961
,4.05,4.050631642678458,2.4195878714468466,4.06,3.5287990997272463,2.1
26512523590597,4.07,3.46133889348159,2.161993328019015,4.08,3.85184357
66214947,2.4761031869179164,4.09,3.162622651179539,2.136632718948933,4
.1,3.3966620250329767,2.357034476069543,4.11,3.181178202335717,2.24775
130230936,4.12,3.402757519430781,2.7724822933788653,4.13,3.42252402081
31766,2.409724368465312,4.14,3.204436957944081,2.6074769679191965,4.15
,2.9963209456950035,2.418362927396508,4.16,3.038603292339582,2.6439683
720760683,4.17,2.664425080649296,2.7474822052014107,4.18,2.79802097794
0009,2.8400688014644473,4.19,2.730844931663201,2.893204583071302,4.2,2
.55099337228026,2.702419876277637,4.21,2.826480288331819,2.78274119965
30698,4.22,2.656237488269432,2.639708139695797,4.23,2.5777608797602243
```

,2.6401538772962923,4.24,2.5616617761051708,2.895649014175869,4.25,2.5
88346917950694,2.577446297111005,4.26,1.9677067424468513,3.06143141177
90657,4.27,2.6267669314228788,2.6038489674076932,4.28,2.19412116354954
2,3.190525150012365З,4.29,2.424808687914503,2.4974502412061628,4.3,1.8
873246678799347,2.9296811541484438,4.31,1.854781178633124,2.8874986876
52754,4.32,2.202945881267647,2.6510158438151663,4.33,2.375764335583012
7,2.549321724059296,4.34,1.9086743845824017,2.841926237113323,4.35,1.6
146363896867986,3.0115345499616484,4.36,1.7150333019397208,2.862595652
90055,4.37,1.878109135517101З,3.0101690931150897,4.38,1.46947680478132
87,3.0567728709614004,4.39,1.6430665479981512,3.022304806794602,4.4,1.
539038834312233,2.8847666401241803,4.41,1.6222487679014808,2.943331891
464817,4.42,1.6127474802004171,2.8158194570467643,4.43,1.3491602251568
74,2.6159511350908935,4.44,1.2799143696078998,2.4774825781392593,4.45,
1.1558415570213882,2.6371623218681592,4.46,1.213404234783244,2.6282978
316408436,4.47,1.3671928120407704,2.5805123265688,4.48,1.1983301906747
053,2.9075211631240556,4.49,0.9347152216288632,2.827413318945958,4.5,1
.29130376694084,3.0137841280185835,4.51,1.3237587377958506,2.569666596
1398404,4.52,1.2273480094272555,2.656288472499893,4.53,1.2602915829470
46,2.438391285878108,4.54,1.2219293610235804,2.471931415108058,4.55,1.
2205978331152472,2.485961809186958,4.56,0.9990833908651997,2.393729049
0104743,4.57,0.9933421400224289,2.364003837423574,4.58,1.0373762332655
379,2.447549360584254,4.59,1.0282243154016986,2.189189958447827,4.6,0.
9382458917959419,2.485970853564102,4.61,0.8019865288745573,2.453888077
6290264,4.62,0.30803727758256816,2.7553002156923947,4.63,0.96651351227
66885,2.176071521262452,4.64,1.2185959128307313,2.721332023667635,4.65
,1.2707359221558572,2.148375680284601,4.66,0.8622059881877602,2.452712
3241593634,4.67,0.7704664501326227,2.140194371387059,4.68,0.9743907663
59058,2.1717165390916326,4.69,0.5836294316445003,2.713852208290541,4.7
,0.913898394100734,2.324883577269828,4.71,0.5938593160829526,2.4745005
338456543,4.72,0.9024738540536684,1.89458166647287,4.73,0.802214674159
6531,1.8893049585710666,4.74,0.7696907749893409,2.1550967572535575,4.7
5,0.4507743753168039,2.152306633587855,4.76,0.8144798972905711,1.89038
74406751375,4.77,0.8799299701028523,1.921629255172585,4.78,1.040257033
5736194,1.9722463131155705,4.79,0.4869929638151541,1.6359058989781552,
4.8,1.0610994619132197,2.181709688831878,4.81,0.42648325402027065,1.89
70792736029807,4.82,0.758451726882917,1.7748071245058754,4.83,0.968204
8228451154,1.8491691767874114,4.84,0.4863067117815585,2.25307729513484
3,4.85,1.0893157108120992,1.3438003689872142,4.86,0.611855067471901,1.
4061076065791824,4.87,0.7062549248949243,1.929422280064047,4.88,0.6611
091155204877,1.9560388120490009,4.89,0.5993213407223166,1.747191277261
4115,4.9,0.9478914844379714,1.8919290235045452,4.91,1.0796883122588983
,1.5868208863031956,4.92,0.7451041688307086,1.5612109644644867,4.93,1.
1183972146454915,1.6428524954690251,4.94,0.93824869439401,1.9740739467
54861,4.95,0.9014052606802417,1.7340297012376928,4.96,0.73637756443253
6,1.4815555963577116,4.97,0.8387197691565456,1.5277980428262876,4.98,0
.7150432899446434,1.7984542153998362,4.99,0.6200096223131559,1.3919056
765343851,5.0,0.5361235339181321,1.723885379318318,5.01,0.872685920859
7127,1.2297672819752408,5.02,0.9292774138724446,1.063888361518229,5.03
,0.960292598442787,1.4055432347636307,5.04,0.7013186961175458,1.212681

```
5668409818,5.05,0.5208745325456842,1.386326767096931,5.06,0.6533200095
112119,1.4751456961620306,5.07,0.7960581593540206,1.3137269339561388,5
.08,1.2784046243503888,1.1131555416744785,5.09,0.7623885136516421,1.59
17949289657338,5.1,1.0597002378471456,1.193240495545422,5.11,1.0365311
71991049,1.8174471883237269,5.12,0.7910081406151375,0.9774377015583389
,5.13,0.7640861444877242,1.3151669692684367,5.14,0.7414844586834907,1.
158013442317809,5.15,0.7766179566342205,1.0850686119218862,5.16,0.9647
71787347673,1.0855928766485716,5.17,0.5822500964782602,1.3563958906407
618,5.18,0.9020117198039991,1.287822773212291,5.19,0.631633437743681,1
.3799705939520828,5.2,0.7526181241537703,1.064566808803256,5.21,0.6715
487255201282,1.1437962997081574,5.22,0.5892371132905683,1.214667597818
921,5.23,0.5111152704125717,1.0191913569928128,5.24,1.0597754643543287
,1.150871439399244,5.25,0.794642153637397,1.0639980880060063,5.26,0.78
20383966852902,1.210938244550694,5.27,0.7196747996902427,0.92988832402
46317,5.28,0.6121158983679689,0.8980966170811113,5.29,0.58732152206176
49,1.3107345959872037,5.3,0.6075916503778449,0.903769583226655,5.31,0.
6845572535372868,0.713919464610707,5.32,0.6493131795622511,0.858820120
2832402,5.33,0.7769434414479653,1.0338347537353734,5.34,0.893597084178
7048,1.0827994434944772,5.35,0.9363229179479109,1.032040928769536,5.3
6,0.5944445803123433,0.7569051844217165,5.37,1.1084819324075559,0.7494
147918152217,5.38,0.831615752875418,0.9040778105321979,5.39,1.11503169
97072227,1.2197046028782945,5.4,1.112993941520571,1.0436761183688117,5
.41,0.9538414773558136,1.0524962034483938,5.42,0.9901681432325111,1.21
70168045901086,5.43,1.0218091860191805,0.7414570911596129,5.44,0.75616
83778527114,0.9465308779304352,5.45,0.7317834845159409,0.8450050362906
94,5.46,0.7285488406762036,0.6604194076712315,5.47,0.9858669631828322,
0.9773083021199316,5.48,0.8541559578756877,1.0290069390794228,5.49,1.0
057320520592394,0.49896267525091265,5.5,1.0163787638737474,0.965508123
6663399,5.51,1.137453072410131,0.91763906234829,5.52,1.110903604739850
4,0.6082629383141988,5.53,0.9433064280958445,1.116980614340847,5.54,1.
091480439363318,0.7445542133354802,5.55,1.0760806235906337,0.640800889
1648653,5.56,1.151756933419378,1.0214210393757637,5.57,0.9727015276852
288,0.768681865352813,5.58,1.1076293233494696,0.4116039038346715,5.59,
1.4444152473044756,0.5384589350275842,5.6,0.9595668406364262,0.6651392
909720962,5.61,1.3811224136892999,0.6762880460710119,5.62,1.3147171986
521227,1.1273437009953902,5.63,1.174310962597466,0.7446208020286644,5.
64,1.59587821476033,0.5368596947524179,5.65,0.9787902579173686,0.65293
275163072,5.66,0.9473643055998615,0.697057826019919,5.67,1.53556917359
0489,0.9429040287214342,5.68,1.418057339502064,0.7487631355266169,5.69
,1.394241728653633,0.1820990948575273,5.7,1.6014305790298926,0.8301086
828125703,5.71,1.5103488042483975,0.9623277385372179,5.72,1.3868799750
987908,0.9431973033711061,5.73,1.0409603618103025,0.7274520464612384,5
.74,1.2024910360178866,0.19534259299516632,5.75,1.4856592329485514,0.2
936712972661631,5.76,1.514034747176429,0.5520390184305666,5.77,1.48685
61124406738,0.6599774856211985,5.78,1.2765081931970157,0.8993046110837
403,5.79,1.7672362643434867,0.31236040156362993,5.8,1.3968318898964025
,0.8787771218706963,5.81,1.6664921422806014,0.41215657566561503,5.82,1
.9515929388909639,0.6234747266118978,5.83,1.817003083160308,0.50038752
35655789,5.84,1.7739471323986535,0.5517561897138747,5.85,1.61405859930
```

39462,0.6117862904699151,5.86,1.7759568661700207,0.7792160501646455,5.
87,1.9810176787948846,0.44928321387683956,5.88,1.813715618720429,0.453
83353431329265,5.89,1.9514973870437546,0.8692300655106845,5.9,1.826464
4428173002,0.8199269225505057,5.91,2.017761030786631,0.492792343303972
87,5.92,2.0711618286480045,0.6375204942470138,5.93,1.9714031606686906,
0.47937546769367784,5.94,2.047088857361915,0.7119796669266629,5.95,1.7
489337957885909,0.638558823142584,5.96,2.1265686481759785,0.9306375591
101992,5.97,2.2668362783557447,0.4330459059805433,5.98,1.8205720138987
083,0.27836081204894425,5.99,2.5033056144866483,1.102310947791596,6.0,
1.9256132985945458,0.673546794868149,6.01,1.7759704461369323,0.7222892
500829153,6.02,2.1520634118050217,0.793203086437455,6.03,2.17424839994
0155,0.6156104291255319,6.04,2.599490160951573,1.0010667360099101,6.05
,2.319863472668282,0.9999412499284783,6.06,2.454326506293957,0.8410721
740779269,6.07,2.6978704206843647,0.793314500620999,6.08,2.10348668791
95784,0.6348467484124166,6.09,2.9898336569420874,0.5729224920537103,6.
1,2.5996870247218546,0.31697334757142526,6.11,2.404432780736069,0.4544
193112003823,6.12,2.8806818041716675,0.7760480569524422,6.13,2.6168532
93221694,0.25427971538176614,6.14,2.7938615439475614,0.653169809438654
7,6.15,2.761219297551689,0.5971828015722882,6.16,2.9994144293386347,0.
91169048409709,6.17,3.096999146056038,0.8614952789429677,6.18,3.257329
942891621,0.7082022360134193,6.19,2.837807055584379,0.9169054155328178
,6.2,3.221590864635033,0.851793750138778,6.21,3.062176930437895,1.1299
152978641263,6.22,3.6264800083570408,0.6559215779921159,6.23,3.1889973
454562655,0.9951538589699672,6.24,3.5258565313446932,0.748683392947359
7,6.25,3.6545650717420646,0.6039367927937777,6.26,3.742831167954424,0.
8332269653301866,6.27,3.555431669952468,0.3791265639219643,6.28,3.9461
191467559953,1.0462253610425816,6.29,3.4959238497585656,1.033406963294
5826,6.3,3.643858817278639,0.7593394129282817,6.31,3.6337945288690237,
0.46156428207424943,6.32,3.7911156695712376,0.8230454663226001,6.33,3.
429874811131566,0.9097825687699901,6.34,3.4348281709035637,1.176840123
8551112,6.35,3.5589007955080723,0.9988989638264664,6.36,3.704742572102
5893,0.4753289372454268,6.37,3.853336105082906,0.9643515549566735,6.38
,3.9351336796771212,1.097811503649343,6.39,4.079284101491908,0.9847833
766001689,6.4,4.154203867078577,1.177584302666303,6.41,4.2675401206328
05,1.018716438493289,6.42,4.370335312088968,0.8408400237834339,6.43,4.
024141062398341,1.1208602621318142,6.44,4.543512031431405,1.1472627245
043183,6.45,4.418338019001059,0.9948702964354396,6.46,4.43394604454401
8,0.8036208750035037,6.47,4.066280540688847,0.9781598061257343,6.48,4.
510758359891607,1.1510430763242325,6.49,4.336482862257395,1.2458112762
18456,6.5,4.203810371491464,1.0588947093478605,6.51,4.399641796255633,
1.2561702404393305,6.52,4.443929620308729,1.042684551870484,6.53,4.051
402217865242,1.5135097470728822,6.54,4.264301994532039,1.4043487836933
45,6.55,4.42400345065097,1.4273501441082665,6.56,4.362065176944761,1.4
364514542144846,6.57,4.378437126579591,1.5061066275755113,6.58,4.98459
3307225643,1.3383520791202617,6.59,4.466089894915959,1.708302930437118
2,6.6,4.641758469226222,1.5326529090590546,6.61,4.278163677325334,1.67
6902617645026,6.62,4.312076793614702,1.6347904498321753,6.63,4.3077533
24180273,1.8125038656583694,6.64,4.529807170962841,1.4263897507845702,
6.65,4.578037382430225,1.8438786884208467,6.66,4.696637037723271,1.649

```
9829698625759,6.67,4.070955688896846,1.7064184030709948,6.68,4.5721029
50041709,1.9247819695152772,6.69,4.865487583483803,1.449361603555952,6
.7,4.296309870555919,1.9487973074769387,6.71,4.324451428371137,2.19873
21097713393,6.72,4.068943119397223,1.9869894834395367,6.73,4.036911637
041486,2.6002164649643125,6.74,3.949408174554147,2.1343094776778,6.75,
4.127391964771095,2.5369371796258853,6.76,3.667010797323429,2.22300161
54807886,6.77,3.922165234347597,2.5643294955074256,6.78,3.720692713576
1747,2.3770803373450424,6.79,3.784513375842681,2.6368274868006827,6.8,
3.622203923270634,2.343526165632942,6.81,3.4073901615408486,2.37471095
08445114,6.82,3.6134792381442153,2.481390493066757,6.83,3.123295660480
929,2.6429493453022785,6.84,3.1587261087870804,3.074135094453306,6.85,
3.256512679299161,2.4334264158021965,6.86,3.197506822019016,2.55828338
1071857,6.87,3.3791741040044685,2.7686041813448026,6.88,2.866647358741
71,2.540726225121887,6.89,3.1209017686962266,2.4566904484098537,6.9,3.
476401336757743,2.7437663628082043,6.91,2.94177837378613,2.71122796024
4224,6.92,2.8238720995410382,2.787336752429913,6.93,2.9381960996553205
,3.1093896753481918,6.94,2.7460674060986037,2.3653109969734603,6.95,2.
7760025826389683,2.9988254450936815,6.96,2.5661453116761663,3.19109077
28722346,6.97,2.8135488739022603,2.749773914652922,6.98,2.316990947005
4746,3.1282661762519552,6.99,2.4714993520845043,2.9788618651177887,7.0
,1.8892601659738388,3.0102512733700237,7.01,2.346277436487754,2.986577
179386613,7.02,2.2731406214443104,2.667218564475599,7.03,2.28304175302
8914,2.7763071396508656,7.04,2.1900933050252305,3.14240675082586,7.05,
1.9038092260166346,3.333092309410554,7.06,1.9428173110601596,3.2928672
697909067,7.07,1.7821693837011927,3.140712955958341,7.08,1.25788713503
18694,3.25384653832248,7.09,1.6712976656390914,3.1077656420489097,7.1,
1.9698020274875223,3.456946819658307,7.11,1.7792149731586133,3.1060421
415446644,7.12,1.5803535704382388,2.9391309287606373,7.13,1.5130816122
21463,3.1049552164244325,7.14,1.2347387681413717,2.952027186837373,7.1
5,1.2977198275796735,2.925409274775479,7.16,1.7404028972201024,3.12619
9931167793,7.17,1.2168642258967273,3.211929728093949,7.18,1.3094759331
85277,2.844932575977484,7.19,1.331022122808068,2.889737929848086,7.2,1
.1067468395909885,3.1034195284400514,7.21,1.372620444517154,2.46293334
53197573,7.22,1.2913770174412238,2.8261626113030296,7.23,1.00023375192
26427,2.8995413702119515,7.24,1.1083228949825792,2.58421608370507,7.25
,1.2576118628002568,2.5689804753713696,7.26,1.328580137692792,2.781713
722958819,7.27,1.1433725047688883,2.498224668993722,7.28,0.96225234152
06011,2.6953881508537014,7.29,1.1284103188900036,2.591766112639343,7.3
,1.343695291294453,2.573848731965955,7.31,0.7214091593082798,2.4692618
047054795,7.32,1.0156953562943811,2.4832884664321466,7.33,0.9299475368
453749,2.4623163885974892,7.34,0.9340655832299127,2.299203880040855,7.
35,0.8152552996620309,2.3629505444732173,7.36,0.7018516185929318,2.594
6052114112224,7.37,0.9045720270755293,2.055455277740282,7.38,0.7485894
569947255,2.1662111210729487,7.39,0.45190721681458823,2.51683606143231
7,7.4,0.7280570240117925,2.1827428159204643,7.41,1.2015167837722454,2.
4095480411553463,7.42,0.816791942803274,1.9911839836866854,7.43,0.6451
72604629921,2.0099474581830465,7.44,0.9125715404530598,2.1350839405572
577,7.45,0.5968969142510594,2.134197138470761,7.46,0.8731833941243968,
2.035901287926412,7.47,0.9315541661785152,2.1892724463165814,7.48,0.49
```

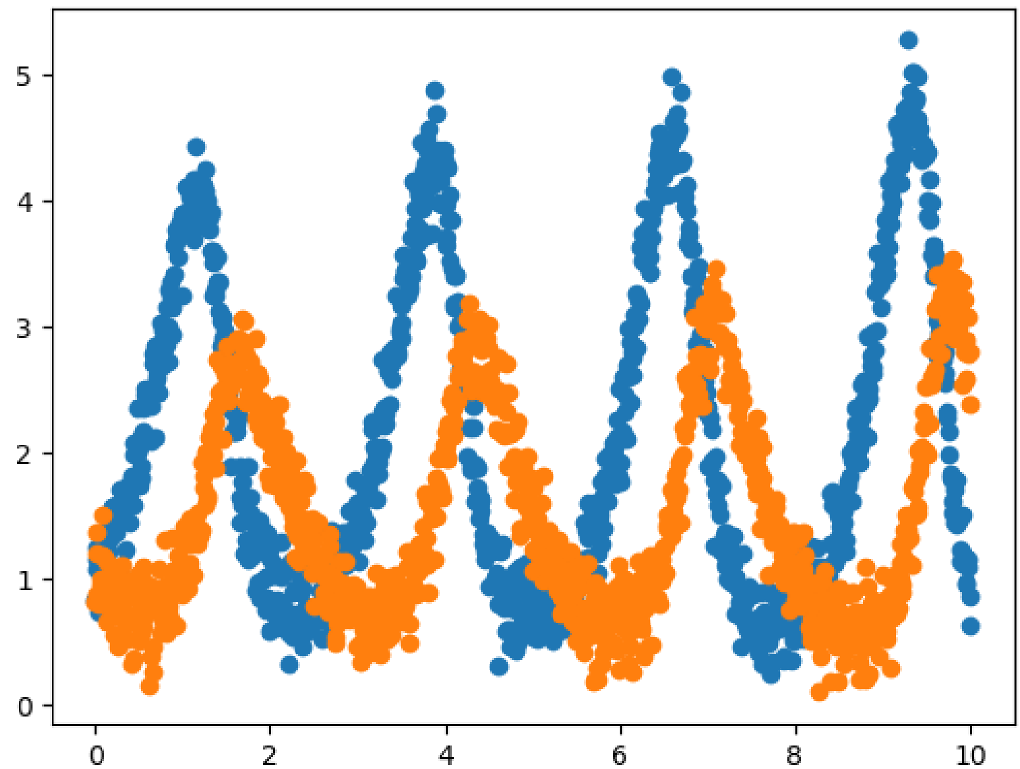
724546689908955,1.9651256077020558,7.49,0.9079230762270225,2.169367553
4571857,7.5,0.6570877041657597,1.8904130719431789,7.51,0.8735970228393
604,1.923848196049785,7.52,0.643395203495106,1.876519050245026,7.53,0.
6349640114082002,1.8729876884487828,7.54,0.6802160271578183,1.77083743
51184846,7.55,0.5201172375381796,1.9562511727926943,7.56,0.58101825679
00222,1.678801871449407,7.57,0.6631642048105172,2.2725188298856667,7.5
8,0.6891802416704929,2.115541277917919,7.59,0.41368688364639217,1.8637
937440606884,7.6,0.6299040463889645,1.7732905700566877,7.61,0.88774141
93350196,1.3934537211007911,7.62,0.31607500331266564,2.047449804196783
,7.63,0.694450935370715,1.8014784838044493,7.64,0.6748713743171243,1.5
23928204911613,7.65,0.7403216649912088,1.6825794469702617,7.66,0.65125
31509926597,1.5123243989120758,7.67,0.663792400504821,1.40103453568530
07,7.68,0.7016519150403998,1.5912560844742336,7.69,0.9126573139518128,
1.0264362245034184,7.7,0.49304116025639755,1.5221715330266443,7.71,0.5
490576039368398,1.3888842306801625,7.72,0.2496542295835017,1.369492060
7959076,7.73,0.3284381795209675,1.2869064861783448,7.74,0.875542413759
9437,1.331517012829524,7.75,0.62898485298962,1.5496945065980412,7.76,0
.6030623949605258,1.5203433969718203,7.77,0.7039237227039769,1.3050389
407751393,7.78,0.6090261778271554,1.627067295513269,7.79,0.74517583648
96471,1.5699642148300055,7.8,0.5968928455364384,1.3536462885840508,7.8
1,0.792596665553277,1.29899814005373,7.82,0.605390449890906,1.09319236
35564718,7.83,0.7230349870210635,1.4221025380623484,7.84,0.63955695511
98461,1.3475936691158958,7.85,0.6898427759012249,1.1776360177278535,7.
86,0.3488169698946642,1.3220380659947297,7.87,0.9205465775379815,1.087
6026639033958,7.88,0.37752205069389144,0.9643405245684533,7.89,0.93717
22243551328,1.1008880621459132,7.9,0.75107036818583,1.0339836667195406
,7.91,0.8480104903625121,0.9677240315237008,7.92,0.6212521774962639,1.
2296860123246576,7.93,0.7861918872101072,0.7542378448324096,7.94,0.590
8252970351764,1.1310582469350738,7.95,0.34338268399780036,0.9816249217
586619,7.96,0.6393202017543316,1.2735705446040375,7.97,0.8557971843449
179,1.121670000743662,7.98,0.5378631075367646,1.3719899054142044,7.99,
0.5101877740864723,1.365878143300695,8.0,0.6668104508918059,0.83263703
21271,8.01,1.2066795692914738,0.8987276222673521,8.02,0.98583407095731
37,1.0230746075748633,8.03,0.6614178541204807,1.0711090194640087,8.04,
0.9785243825752304,1.324725895839951,8.05,0.9550367413023738,1.1283756
790819324,8.06,0.7185310698281548,0.9229915904167277,8.07,1.2591417872
376585,0.7838700760867008,8.08,0.7789940853257453,0.9570395093725057,8
.09,0.5126518145727842,1.3730608728150817,8.1,1.040809265291164,1.0703
329607245817,8.11,0.7170786617265582,0.863956399279386,8.12,0.79696981
41711359,1.180490390536049,8.13,0.6602428600552204,0.9754792751515664,
8.14,1.1281337672972778,0.91523978858007,8.15,0.9443989531412406,0.616
2455876034539,8.16,0.7938549569713079,0.8428965649936907,8.17,1.068935
7134317448,0.49922114030901216,8.18,1.26618462400718,0.553178187901359
6,8.19,0.8984207327542956,0.6695427202178923,8.2,1.0151321537999543,0.
8178766674600964,8.21,1.0274980239724745,0.7825575138407148,8.22,0.777
9249799865273,0.7958936351915602,8.23,1.0214200268807303,0.52514250122
08515,8.24,0.7119114046609646,0.9671082700311512,8.25,1.24217662111462
1,0.6102948445047187,8.26,1.0859310833867717,0.09659526432822652,8.27,
1.0129712278676462,0.488798385206379,8.28,1.0454577667812042,0.4869778
9574137423,8.29,0.7103710985321137,0.6567457498727434,8.3,0.7222130880

389985,0.38446302120245934,8.31,0.8737418677519565,0.9611357333326054,
8.32,1.0876124119509711,0.8648022170423788,8.33,0.9927546071495009,1.0
670043674090297,8.34,1.06274257055628,0.6594319283203098,8.35,0.991895
3405753131,0.5499112851565217,8.36,0.9156422837565699,0.82095259326953
63,8.37,0.7533330943279255,0.6326307642417249,8.38,0.9271104264490396,
0.6278300804894646,8.39,1.021664475121458,0.7553380023743694,8.4,1.679
4080348607698,0.18234908924984428,8.41,1.2683631666908213,0.6639309105
165161,8.42,1.5800177509714743,0.7492370787751281,8.43,1.6957406523895
25,0.5217465663966968,8.44,1.1416714444050498,0.5042734854569559,8.45,
1.1174686831191205,0.7177849155143045,8.46,1.161147271127489,0.7809286
079999692,8.47,0.8805496061167629,0.4512415666676035,8.48,1.5088001328
896117,0.9171666682417565,8.49,1.3146079736140672,0.8770243440101686,8
.5,1.2148975441618923,0.18072722224186355,8.51,1.4790121816641004,0.92
12881419669594,8.52,1.4121981611522727,0.7160501312794041,8.53,1.65331
50397495417,0.5073864777246748,8.54,1.3977274992824018,0.7258224192498
253,8.55,1.5436568652138276,0.6132409227101607,8.56,1.4462762630134616
,0.3250446682530695,8.57,1.2063064714012257,0.42464549732892365,8.58,1
.6555232010107268,0.46182151731554594,8.59,1.3073426324013053,0.704716
9209498931,8.6,1.7707402325538255,0.5615960784346329,8.61,1.7336065187
20465,0.8687756288366092,8.62,1.815536514102736,0.46935489081816434,8.
63,2.0042354249938437,0.507500697560649,8.64,1.9338439056195496,0.8833
894793963066,8.65,1.6123615533839688,0.46688014281033574,8.66,2.048115
9441789796,0.8866024619781667,8.67,2.0057649929461236,0.56192147210121
35,8.68,2.0023322713725977,0.635754768665086,8.69,2.3561941547900265,0
.730988767964013,8.7,2.2232541859750765,0.7359733960573609,8.71,2.1374
696697573357,0.7282076148711497,8.72,2.00929051521747,0.58420650986148
93,8.73,2.051754591384316,0.3347272255535666,8.74,1.92414223252858,0.1
906663631698927,8.75,2.274220591014234,0.6154724585732794,8.76,2.5329
92989529698,0.47205446310089566,8.77,2.574396894869018,0.4355966865433
8046,8.78,2.4094857380064307,0.6292866719719764,8.79,2.158489714647318
5,0.5263187374709929,8.8,2.3862847247759738,0.1947392476790058,8.81,2.
5641975438030005,1.0950748682752136,8.82,2.9301186539958173,0.49691984
401004835,8.83,2.1175639641459,0.9333261198924512,8.84,2.4270919685027
94,0.24326156358588147,8.85,2.5132151773550206,0.6311818369350817,8.86
,2.6890984213013716,0.8022658796607038,8.87,2.7763731494798503,0.71640
25943683793,8.88,2.6752779163584854,0.8329711440146121,8.89,2.63458638
60501857,0.5259951316226059,8.9,2.9576842675520165,0.4837109454829116,
8.91,2.819284174778445,0.8070422681854045,8.92,3.2821717747114603,0.69
41711151696986,8.93,2.9730143070346644,0.716237274185198,8.94,3.593473
1662541775,0.5995747705448086,8.95,3.329742950111208,0.781998170745491
4,8.96,3.4178268462859522,0.37622097796370146,8.97,3.3395568600436083,
0.7047797130192099,8.98,3.1558214925393155,0.6482030208887144,8.99,3.4
91036793053872,0.7723879211815173,9.0,3.5418229588546195,0.65209182001
25979,9.01,3.4512364713814483,1.0353969133087313,9.02,3.72941933064726
9,0.8345640825888871,9.03,3.8418820864703327,0.7001637786533615,9.04,3
.516878545875596,0.5870593210924844,9.05,3.4148451843786964,0.52391824
63607857,9.06,3.639794878408449,0.7250000453056216,9.07,4.152739345421
453,0.6965661314733301,9.08,4.048070883233181,0.8724216801144075,9.09,
3.924829268587241,1.0035487225807205,9.1,3.8175578675402777,0.28335826

04066477,9.11,4.19412720357935,0.6916807360699069,9.12,3.982291866192756,0.8166123432484306,9.13,4.236311393981772,1.034540658958018,9.14,4.322917660018796,0.8553966670119082,9.15,4.258639406605894,0.8656391798004786,9.16,4.608423771261483,0.7239566190150666,9.17,4.619315966642123,1.0127404271979232,9.18,4.55763431061662,0.7728146895803263,9.19,4.144602951747909,1.280966190474749,9.2,4.277996836615191,1.111795925996073 4,9.21,4.45886292441497,0.999892360237856,9.22,4.460793600177539,0.8901229604196672,9.23,4.405303104272578,0.9553520441284657,9.24,4.732059248288891,1.1632662323337015,9.25,4.4678497716736345,1.3808989028655847,9.26,4.72485076283413,1.503637582319569,9.27,4.371465750986808,1.2833243314137055,9.28,4.754089977621212,1.3366426301764824,9.29,4.709217943072764,1.3830843363782137,9.3,5.274566562538437,1.4551450244210329,9.31,4.8723257397874935,1.1190119520843804,9.32,4.763849791663451,1.3157009976094731,9.33,5.025923387379227,1.099359940082938,9.34,4.588455341250027,1.7266269622473418,9.35,5.021264492565694,1.7555649084812306,9.36,4.7802488163852095,1.6983924732518223,9.37,4.820107249375715,1.7850885981788387,9.38,4.80353679281785,1.5085226082673417,9.39,4.651234019697264,1.5686034179069686,9.4,4.531191549819501,1.5270014800245468,9.41,4.988543820426855,2.021806533408517,9.42,4.568184354084061,1.844032499788395,9.43,4.479102438359191,1.8842076319483672,9.44,4.367243778317708,2.038953797110904,9.45,4.328986156009474,1.9895340724269397,9.46,4.373303238081175,2.3244479875804296,9.47,4.445614172918336,2.2026031668385655,9.48,4.361866109303184,2.523527917148253,9.49,4.380518546690104,2.0148235895382354,9.5,4.381401628216147,1.9909667302762042,9.51,4.003752985803329,2.2240474225779168,9.52,3.8802650094347726,2.240719248278699,9.53,3.8427171626011747,2.83513544986062,9.54,4.1791085989599175,2.2307712092359875,9.55,3.993308536371128,2.541496585527653,9.56,3.569907940415251,2.6046739904410416,9.57,3.645188473888739,2.609519057912961,9.58,3.4011769241034644,2.59871191254008,9.59,3.56781388598903,2.837722503093758,9.6,3.508364928823892,3.1422760740270963,9.61,2.8641817154771005,2.6588152031981553,9.62,3.041047256709463,2.804405774507134,9.63,2.7662876556997635,3.4113695440787133,9.64,3.1761868986611517,2.9244451795162343,9.65,2.9550948462233286,3.1839234566685315,9.66,2.885657925067238,2.7875273172368633,9.67,2.820352967397442,3.243064385668716,9.68,2.772423409589187,3.1464505729026224,9.69,2.740429589139801,2.945380360224359,9.7,2.8081562357784673,3.1976433778504436,9.71,2.560999109898741,3.3302935269579432,9.72,2.627012786091859,2.931849515524009,9.73,2.3166608235297064,3.442414243493739,9.74,2.2777299553169255,3.0678926126609354,9.75,2.173300300510063,3.39922562880594,9.76,1.9777776367216078,2.962584761951141,9.77,1.8341924238024478,3.233129544857666,9.78,1.7939413740208932,3.275484253617216,9.79,1.5277042905347673,3.0909296553848447,9.8,1.6152850101238374,3.5389830543524754,9.81,1.7890234746273086,3.5218656513563653,9.82,1.4466524831361462,2.9125387784377303,9.83,1.7108383750799188,3.065106212789417,9.84,1.7761125340405224,3.289666520493026,9.85,1.430253693891255,3.3843648274838594,9.86,1.4689077544364568,3.198566360837759,9.87,1.4534505944228073,2.9807391200121747,9.88,1.1390487558799256,2.8899445855388497,9.89,1.2342653208037062,3.121572496885211,9.9,1.505032691485468,3.0911022380676356,9.91,1.2044135753527119,3.3614025173351845,9.92,1.114576051774367,2.534184578106

```
719,9.93,1.1746687763543184,2.795936310726534,9.94,1.1678430675376203,
3.2101133489501694,9.95,0.9602510691145131,2.898149790878264,9.96,0.95
86668177954598,2.579853795166528,9.97,1.1489916679787533,2.78070546392
66764,9.98,1.0962140845023651,3.0805176189732597,9.99,0.63554526049440
31,2.796228963036231,10.0,0.8571590506095471,2.3893746095481205]).resh
ape((-1,3))

plt.scatter(data[:, 0], data[:, 1])
plt.scatter(data[:, 0], data[:,2])
```

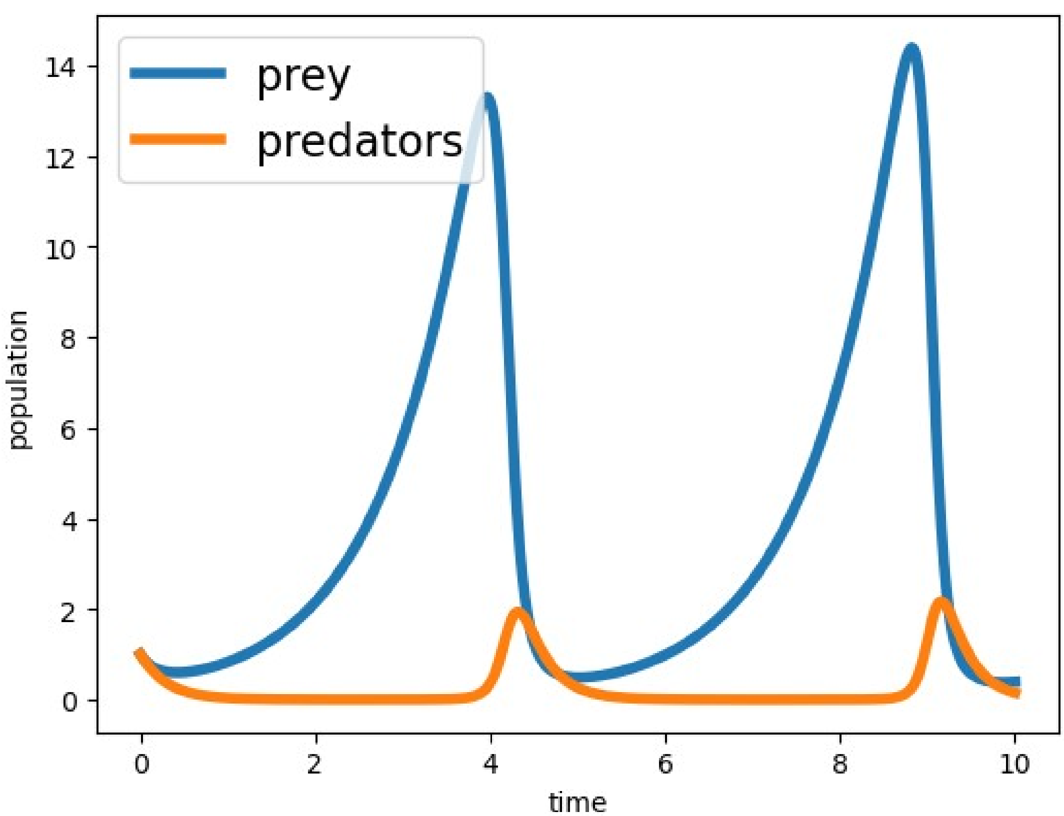
```
<matplotlib.collections.PathCollection at 0x23250c21a10>
```



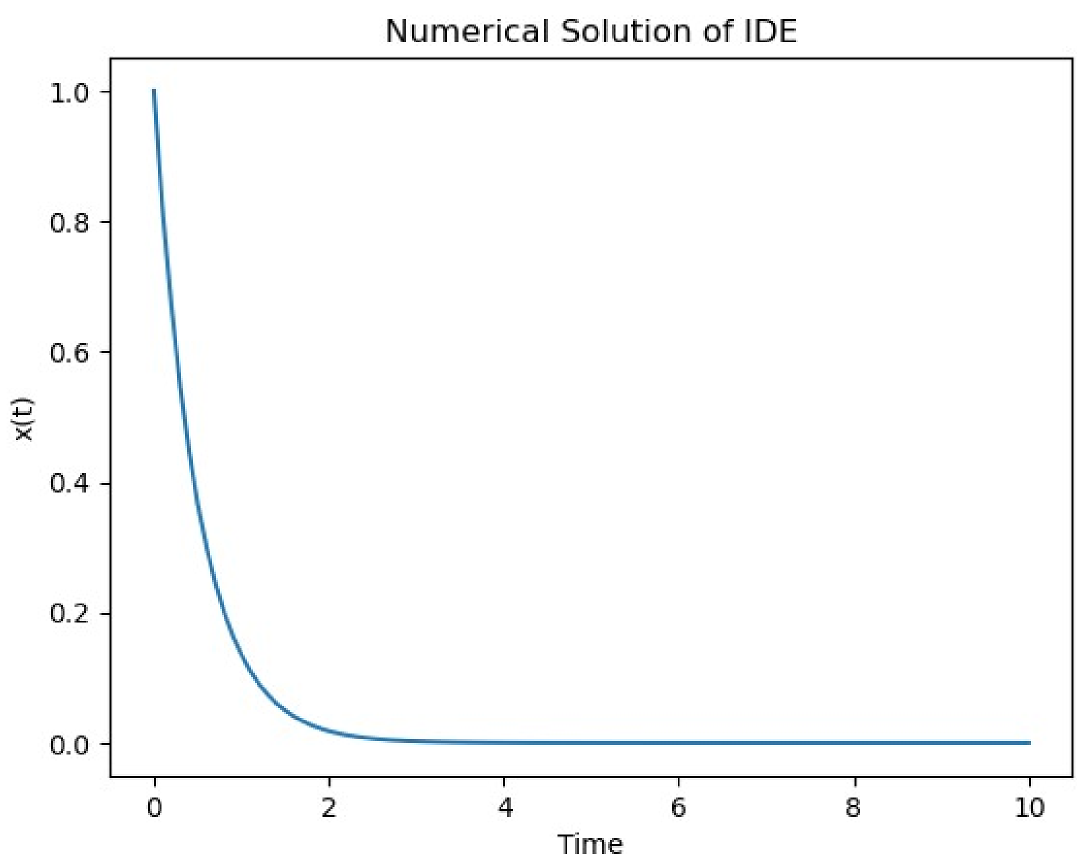

```
def parameterise_basic_model(p):
    a, b, c, d = p

    def model(x, t):
        N, P = x
        return [a * x[0] - b * np.prod(x),
                c * np.prod(x) - d * x[1]]

    return model


basic_model = parameterise_basic_model([1, 4, 1, 4])
```

```
def solve(f, tspan, x0):
    δt = 0.01
    ts = np.arange(tspan[0], tspan[1] + δt, δt)
    xs = np.zeros((len(x0), len(ts)))
    xs[:, 0] = x0

    for i in range(1, len(ts)):
        xs[:, i] = xs[:, i-1] + δt * np.array(f(xs[:, i-1], ts[i]))

    return ts, xs.T


t_span = (0, 10)
x_0 = np.array([1.0, 1.0])
ts, xs = solve(basic_model, t_span, x_0)

plt.plot(ts, xs[:, 0], label="prey", linewidth=4)
plt.plot(ts, xs[:, 1], label="predators", linewidth=4)
plt.xlabel("time")
plt.ylabel("population")
plt.legend(fontsize=16)
```

```
<matplotlib.legend.Legend at 0x232514f17d0>
```



```python
#solve the differential equation
from scipy.integrate import solve_ivp
#Define the ODE function
def f(t,x):

  return [-2 *x[0]]
#Define time span
tspan = (0,10)

#initial conditions
x0 = [1,0]
sol = solve_ivp (f, tspan, x0, method = 'RK45', t_eval =
np.linspace(tspan[0],tspan[1],100))

#plot
plt.plot(sol.t, sol.y[0])
plt.xlabel("Time")
plt.ylabel("x(t)")
plt.title('Numerical Solution of IDE')
plt.show()
```



4)e)

- If we measure predators and prey in units of a single animal $(let(N'=N) and (P'=P))$, we can rewrite the differential equations using the chain rule.
- Let (N') and (P') be the new variables:

  $$\left[\frac{dN'}{dt}=aN'-bN'P'\right)\left[\frac{dP'}{dt}=cN'P'-dP'\right)$$

- These equations represent the dynamics of the prey and predator populations when measured in units of a single animal.

4)f)

```
#4)f)
import numpy as np

def simulation(p):
    a, b, c, d = p
    dt = 0.01
    num_steps = 1001
    timepoints = np.arange(0, 10, dt)
    populations = np.zeros((num_steps, 2))
    populations[0] = [1, 1]  # Initial conditions

    for t in range(1, num_steps):
        dN_dt = a * populations[t-1, 0] - b * populations[t-1, 0] *
populations[t-1, 1]
        dP_dt = c * populations[t-1, 0] * populations[t-1, 1] - d *
populations[t-1, 1]
        populations[t] = populations[t-1] + dt * np.array([dN_dt,
dP_dt])

    return populations

# Example usage:
parameters = [2, 3, 4, 5]
result = simulation(parameters)

print(result)

# The rest of the code seems correct
simulation_result = simulation(parameters)
print(simulation_result[:, 0])
```

```
[[1.         1.        ]
 [0.99       0.99      ]
 [0.980397   0.979704  ]
 ...
 [1.41092787 1.43386793]
 [1.3784539  1.4430979 ]
 [1.34634566 1.45051276]]
[1.         0.99       0.980397   ... 1.41092787 1.3784539
1.34634566]
```

4)g)

```python
#4)g)
def mse(p):
    simulation_result = simulation(p)
    residuals = simulation_result - data[:, 1:3]
    mse_value = np.mean(residuals**2)
    return mse_value

# Example usage:
parameters = [2, 3, 4, 5]
mse_value = mse(parameters)
print(mse_value)
```

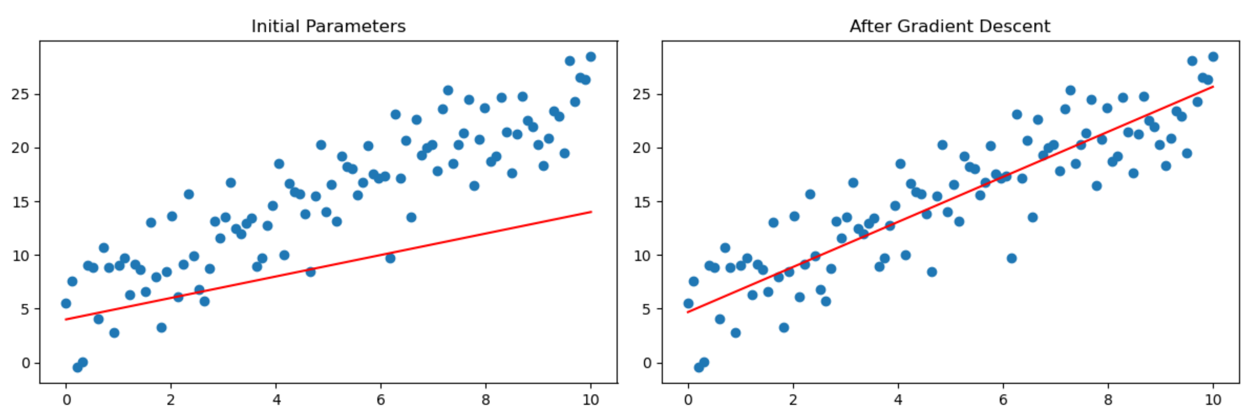
```
1.969430985356843
```

4)h)

```python
#4)h)
import numpy as np
import matplotlib.pyplot as plt

# Generate synthetic data
x_data = np.linspace(0, 10, 100)
y_true = 2 * x_data + 5 + np.random.normal(0, 3, len(x_data))

# Initial parameter values
theta0 = np.array([1, 4, 1, 4])

# Mean Squared Error (MSE) function
def mse(theta):
    y_pred = theta[0] * x_data + theta[1]
    return np.mean((y_true - y_pred) ** 2)

# Gradient of MSE function
def gradient(theta):
```

```
    y_pred = theta[0] * x_data + theta[1]
    error = y_true - y_pred
    g0 = -2 * np.mean(x_data * error)
    g1 = -2 * np.mean(error)
    return np.array([g0, g1, 0, 0])

# Gradient Descent
theta = theta0.copy()
learning_rate = 0.01
for i in range(100):
    theta = theta - learning_rate * gradient(theta)

# Compute MSE for initial and final parameters
mse_initial = mse(theta0)
mse_after_gradient_descent = mse(theta)

# Check if MSE reduced by a factor of 3
if mse_initial / mse_after_gradient_descent >= 3:
    print("Success! MSE reduced by a factor of 3.")

# Plotting
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

# Plot with initial parameters
ax1.scatter(x_data, y_true)
ax1.plot(x_data, theta0[0] * x_data + theta0[1], c='r')
ax1.set_title('Initial Parameters')

# Plot after gradient descent
ax2.scatter(x_data, y_true)
ax2.plot(x_data, theta[0] * x_data + theta[1], c='r')
ax2.set_title('After Gradient Descent')

plt.tight_layout()
plt.show()
```

```
Success! MSE reduced by a factor of 3.
```



```
def gradient(f, x):
    n = len(x)
    δx = 0.01

    def Δx(i):
        z = np.zeros(n)
        z[i] = δx
        return z

    dfdx = np.zeros(n)
    for i in range(n):
        dfdx[i] = (f(x + Δx(i)) - f(x)) / δx

    return dfdx
```

4)i)

In general, achieving an MSE of 0 is uncommon and often suggests overfitting to the noise present in the data. An MSE of 0 implies a perfect match between the model's predictions and the observed values, with no discrepancies. However, this situation can arise due to noise, measurement errors, or other data-related factors.

Consider the scenario where MSE equals 0. In such cases, the parameter values leading to a perfect fit would typically be unique, assuming the model is well-defined. There shouldn't be another distinct set of parameters that perfectly fits the data.

In the context of achieving an MSE of 0, let's consider another scenario to provide an alternative explanation:

Suppose we have a quadratic regression model given by $\left(y=ax^2+bx+c\right)$, and it perfectly fits the data with an MSE of 0. In this case, we can explore another set of parameters $\left(a'\right)$, $\left(b'\right)$, and $\left(c'\right)$ that also perfectly fits the data.

Original model: $\left(y=ax^2+bx+c\right)$

Now, consider a new set of parameters: $\left(a'=2a\right)$, $\left(b'=2b\right)$, and $\left(c'=2c\right)$. The corresponding model becomes:

New model: $\left(y'=a'x^2+b'x+c'=2ax^2+2bx+2c\right)$

Since the original model perfectly fits the data, we can express $\left(ax^2+bx+c\right)$ as $\left(y\right)$. Substituting this into the new model, we get:

$\left(y'=2y\right)$

This implies that any data point (x, y) that satisfies the original quadratic regression model $\left(y=ax^2+bx+c\right)$ will also automatically satisfy the new model $\left(y'=2y\right)$. Therefore, the curves defined by the original quadratic model and the new quadratic model perfectly overlap each other, passing through all data points.

This illustrates that, in the case of achieving an MSE of 0, there can be alternative sets of parameters that result in models perfectly fitting the data, and these alternative parameter sets may lead to equivalent models that perfectly overlap.

Question 5

5)a)

The variable (p) in the nondimensionalized model is defined as the product of the growth rate parameter (r) and the characteristic time scale (T), and it is related to the original parameters of the competition model.

The relationship between (T) and (t) is given by (T = r t), where (r) is the growth rate parameter. If, for example, (t) is measured in seconds and (r) is 60, then (T) would be in minutes.

Now, let's express (p) in terms of the original parameters:

$[p = r T $

Substitute $\left(T = r\,t\right)$:

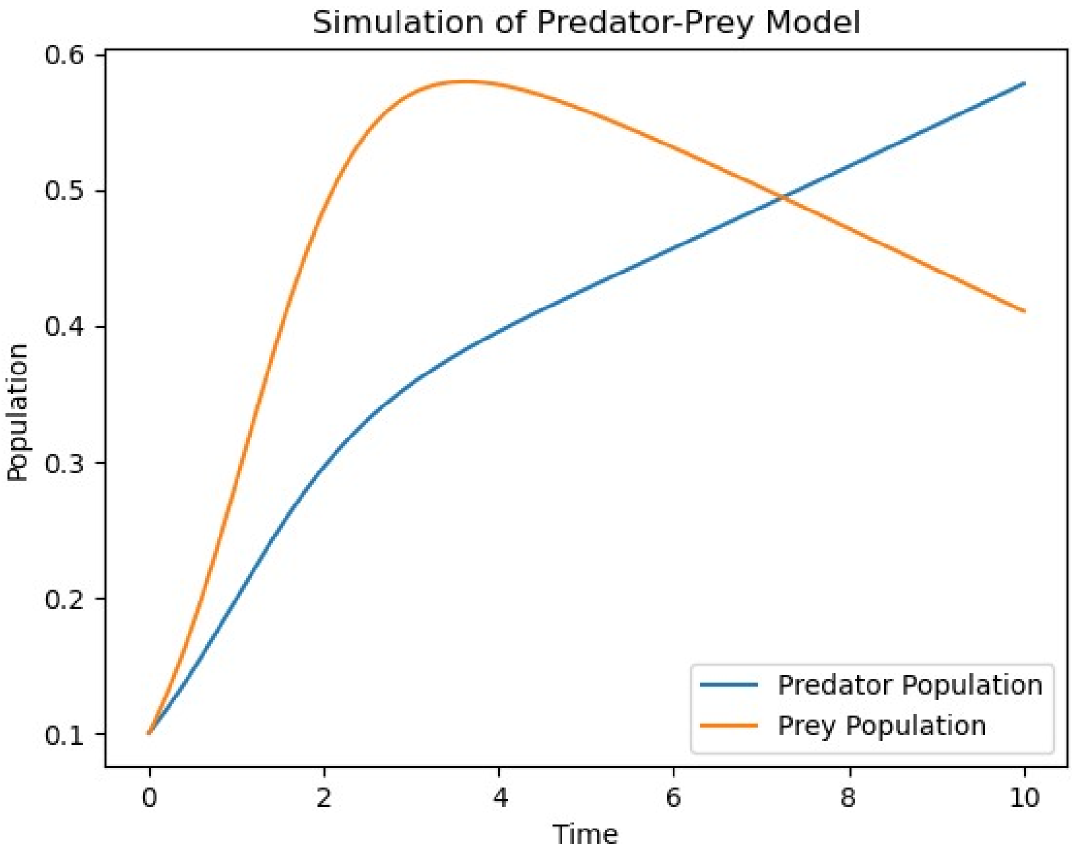
$\left[p = r^2 t\right)$

In the context of the original competition model, (p) represents the product of the growth rate (r) and a characteristic time scale (t). It essentially quantifies the time scale at which the population dynamics evolve. In biological terms, (p) can be interpreted as a measure of how quickly the populations of the competing species change concerning the growth rate (r) and the time (t). A larger (p) suggests a faster dynamics, while a smaller (p) indicates a slower evolution of the populations.

5)b)

```
#5)b)

from scipy.integrate import odeint

# Define the system of differential equations
def predator_prey_model(populations, time, interaction_coeff_1, 
interaction_coeff_2, growth_rate):
    predator, prey = populations
    d_predator_dt = predator * (1 - predator - interaction_coeff_1 * 
prey)
    d_prey_dt = growth_rate * prey * (1 - prey - interaction_coeff_2 *
```

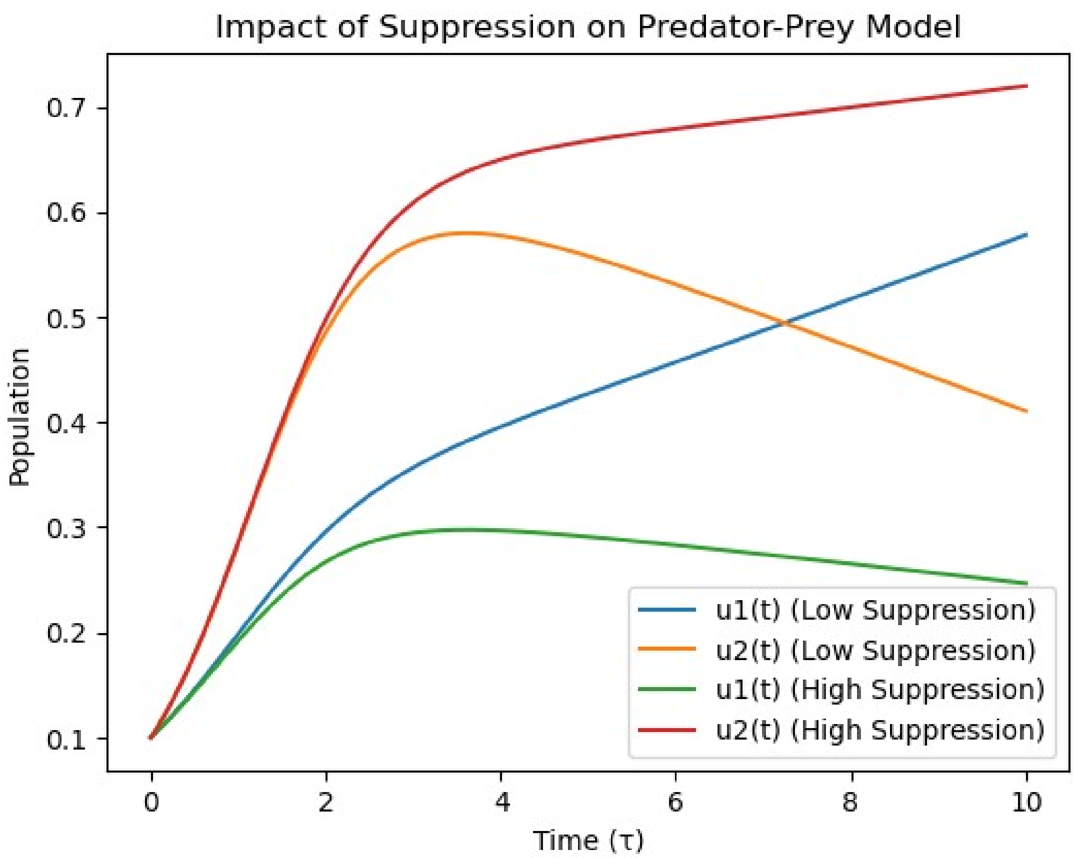
```
predator)
    return [d_predator_dt, d_prey_dt]

# Set initial conditions and parameters
initial_populations = [0.1, 0.1]
interaction_coeff_1 = 0.9
interaction_coeff_2 = 1.1
growth_rate = 1.6

# Set time points for simulation
time_points = np.linspace(0, 10, 1000)

# Simulate the system of differential equations
simulation_result = odeint(
    predator_prey_model,
    initial_populations,
    time_points,
    args=(interaction_coeff_1, interaction_coeff_2, growth_rate)
)

# Plot the simulation results
plt.plot(time_points, simulation_result[:, 0], label='Predator
Population')
plt.plot(time_points, simulation_result[:, 1], label='Prey
Population')
plt.xlabel('Time')
plt.ylabel('Population')
plt.title('Simulation of Predator-Prey Model')
plt.legend()
plt.show()
```



5)c)

```
#5)c)
# Define the system of differential equations
def model(u, τ, a12, a21, p):
    u1, u2 = u
    du1dt = u1 * (1 - u1 - a12 * u2)
    du2dt = p * u2 * (1 - u2 - a21 * u1)
    return [du1dt, du2dt]

# Set initial conditions and common parameters
u0 = [0.1, 0.1]
a21 = 1.1
p = 1.6

# Set time points for simulation
τ = np.linspace(0, 10, 1000)

# Simulate with low suppression (a12 = 0.9)
a12_low = 0.9
result_low = odeint(model, u0, τ, args=(a12_low, a21, p))
```

```
# Simulate with high suppression (a12 = 1.1)
a12_high = 1.1
result_high = odeint(model, u0, τ, args=(a12_high, a21, p))

# Plot the simulations
plt.plot(τ, result_low[:, 0], label='u1(t) (Low Suppression)')
plt.plot(τ, result_low[:, 1], label='u2(t) (Low Suppression)')
plt.plot(τ, result_high[:, 0], label='u1(t) (High Suppression)')
plt.plot(τ, result_high[:, 1], label='u2(t) (High Suppression)')
plt.xlabel('Time (τ)')
plt.ylabel('Population')
plt.title('Impact of Suppression on Predator-Prey Model')
plt.legend()
plt.show()
```



5)d)

To find the fixed points, we set the derivatives of (u_1) and (u_2) to zero:

$$\left[\frac{du_1}{d\tau} = u_1\left(1 - u_1 - a_{12}u_2\right) = 0\right)$$

$$\left[\frac{du_2}{d\tau} = p\,u_2\left(1 - u_2 - a_{21}u_1\right) = 0\right)$$

For (u_1), the solutions are (u_1 = 0), (u_1 = 1), and $\left(u_1 = \frac{1 - a_{12}u_2}{1 - a_{12}}\right)$.

For (u_2), the solutions are (u_2 = 0) and $\left(u_2 = \frac{1 - a_{21}u_1}{1 - u_2}\right)$.

Now, we combine these solutions to find the fixed points:

1. $\left(u_1 = 0, u_2 = 0\right)$
2. $\left(u_1 = 1, u_2 = 0\right)$
3. $\left(u_1 = \frac{1 - a_{12}u_2}{1 - a_{12}}, u_2 = \frac{1 - a_{21}u_1}{1 - u_2}\right)$
4. $\left(u_1 = 0, u_2 = 1\right)$

Interpretations:

1. Both species are extinct.
2. Species 1 dominates, and species 2 is extinct.
3. Coexistence point where the two species coexist in a stable manner.
4. Species 2 dominates, and species 1 is extinct.

These fixed points provide insights into the possible outcomes of the competition between the two species, ranging from coexistence to the dominance of one species over the other. The stability of the coexistence point is particularly interesting, as it represents a balanced state where both species can persist over time.

5)e)

To calculate the Jacobian matrix (J(x)) for the system of differential equations

\begin{align} |frac{du_1}{d|tau} &= u_1(1 - u_1 - a_{12}u_2) | |frac{du_2}{d|tau} &= pu_2(1 - u_2 - a_{21}u_1), |end{align}

we differentiate each equation with respect to (u_1) and (u_2) and the Jacobian matrix is given by:

J(x) = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \end{bmatrix}

J(x) = \begin{bmatrix} 1 - 2u_1 - a_{12}u_2 & -a_{12}u_1 \ p(1 - u_2 - a_{21}u_1) & p(1 - 2u_2 - a_{21}u_1) \end{bmatrix}

= \begin{bmatrix} 1 - 2u_1 - a_{12}u_2 & -a_{12}u_1 \ -pa_{21}u_2 & p(1 - 2u_2 - a_{21}u_1) \ end{bmatrix}.

So,

J(x) = \begin{bmatrix} 1 - 2u_1 - a_{12}u_2 & -a_{12}u_1 \ -pa_{21}u_2 & p(1 - 2u_2 - a_{21}u_1) \ end{bmatrix}.

5)f)

Evaluating the Jacobian matrix at the fixed points:

- (u_1 = 0, u_2 = 0):

J(0, 0) = \begin{bmatrix} 1 & 0 \ 0 & p \end{bmatrix}

- (u_1 = 1, u_2 = 0):

J(1, 0) = \begin{bmatrix} -1 & -a_{12} \ p(1 - a_{21}) & p(1 - 2) \end{bmatrix}

- $\left(u_1 = \frac{1 - a_{12} u_2}{1 - a_{12}}, u_2 = \frac{1 - a_{21} u_1}{1 - u_2}\right)$:

This point is nontrivial to compute analytically.

- (u_1 = 0, u_2 = 1):

J(0, 1) = \begin{bmatrix} 1 & 0 \ 0 & -p \end{bmatrix}

Stability requirements:

For stability, we look at the eigenvalues of the Jacobian matrix. Specifically:

- If both eigenvalues have negative real parts, the fixed point is stable.
- If at least one eigenvalue has a positive real part, the fixed point is unstable.

The requirements are:

1. $\left(u_1 = 0, u_2 = 0\right)$: Stable if $\left(p > 0\right)$.
2. $\left(u_1 = 1, u_2 = 0\right)$: Stable if $\left(p < 0\right)$ and $\left(1 - a_{21} > 0\right)$.
3. Coexistence point: Complicated conditions involving $\left(a_{12}\right)$ and $\left(a_{21}\right)$.
4. $\left(u_1 = 0, u_2 = 1\right)$: Unstable regardless of parameters.

These conditions indicate under what parameter values the fixed points are stable in the predator-prey model.