Assignment: 1

Mathematical Modelling in Industry

Name: **Lokendra Kumar**
Roll No: MA23M008

Submitted to: Dr. Sundar S
Date of Sub: 21/08/2023

1 : Assignment

Q.1 How many cherries each of radius r can be packed in a can of radius R and height h? Obtain upper and lower bounds.
Ans. For upper bound of cherries we will use The face-centered cubic (FCC) lattice packing. It is a well-known arrangement for spheres in three-dimensional space that achieves one of the highest packing densities among regular packing arrangements. The FCC lattice packing involves arranging spheres in such a way that they form a cubic lattice with additional spheres placed at the centers of each of the six faces of the cube.

Packing Fraction Calculation: To prove the efficiency of FCC packing, we can calculate the packing fraction, which is the ratio of the volume occupied by the spheres to the total volume of the arrangement.

Let:
r = radius of each sphere
h = height of the can
R = radius of the can
The volume of each sphere is $V_{\text{sphere}} = \frac{4}{3}\pi r^3$.
The volume of the can is $V_{\text{can}} = \pi R^2 h$.
Since there are eight corner spheres in each can, the total volume occupied by spheres in the can is $8 \cdot \frac{4}{3}\pi r^3$.
The packing fraction (ϕ) is:

$$\phi = \frac{\text{Total volume occupied by spheres}}{\text{Total volume of the can}}$$

$$\phi \approx 0.74048$$

The result, approximately 0.74048, is the packing fraction achieved by the FCC lattice packing. This value corresponds to the fraction of space within the unit cell that is effectively occupied by spheres. The FCC lattice packing achieves a relatively high packing density, making it one of the most efficient regular packing arrangements for spheres.

Upper Bound: The maximum number of cherries that can be packed into the can is limited by the volume of the can and the volume of a cherry. We can calculate the volume of the can as a cylinder and the volume of a cherry as a sphere.

Upper bound: Max cherries upper $= \frac{V_{\text{can}}}{V_{\text{cherry}}}$
However, this upper bound is not very tight, so we can refine it further:

$$\text{Max cherries upper refined} = \frac{0.74 \cdot V_{\text{can}}}{V_{\text{cherry}}}$$

$$= \frac{0.74 \cdot \pi R^2 h}{\frac{4}{3}\pi r^3}$$

Lower Bound: In this arrangement, each sphere is surrounded by six others in a hexagonal pattern, but the spheres are not as closely packed as in hexagonal close-packing.

1. Calculate the area of the base of the can:

$$A_{\text{base}} = \pi R^2.$$

2. Calculate the area occupied by a single cherry at its base:

$$A_{\text{cherry}} = \pi r^2.$$

3. Calculate the maximum number of cherries that can be packed in the base:

$$n_{\text{base}} = \frac{A_{\text{base}}}{A_{\text{cherry}}}.$$

4. Calculate the maximum number of layers that can be stacked in the height of the can:

$$n_{\text{layers}} = \frac{h}{2r}.$$

5. Calculate the lower bound for the number of cherries:

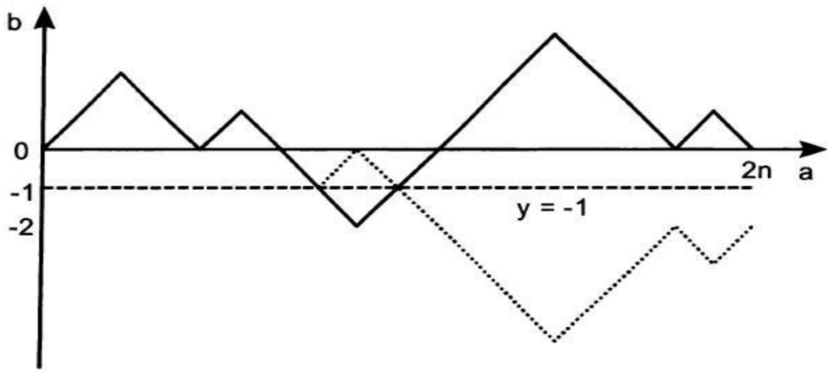
$$n_{\text{lower}} = n_{\text{base}} \times n_{\text{layers}}$$

$$n_{\text{lower}} = \left(\frac{\pi R^2}{\pi r^2}\right)\frac{h}{2r} = \frac{R^2 h}{r^3}.$$

Q.4 There are $2n$ people lined up at a ticket office: n have only 5 bills; n have only 10 bills. The box office has no cash when it opens, and each customer will purchase one 5 ticket. What is the probability that no customer waits for change? For $n = \frac{k\pi}{2}$ estimate the answer using Monte Carlo simulation, where k is the last non-zero digit of your roll number.

Ans. We can utilize the reflection principle to solve this problem. This particular problem exemplifies one of the numerous scenarios in which possessing extensive knowledge proves advantageous. We assign a value of +1 to the n individuals holding \$5 bills and -1 to the n individuals possessing \$10 bills. We can envision the progression as a walk, and denote the state after a certain number of steps as (a, b).

- Start at point $A = (0, 0)$.
- Every time a \$5 person wants to buy a ticket, you move one unit to the right and one unit upward.
- Every time a \$10 person wants to buy a ticket, you move one unit to the right and one unit downward.



Reflected paths: The dashed line is the reflection of the solid line after it reaches -1.

This way, after serving all $2n$ individuals, your journey originates from A and culminates at a specific point $B = (2n, 0)$.

The count of all possible pathways is simple to determine:

Total pathways, $N_{\text{total}} = \binom{2n}{n} = \frac{(2n)!}{n! \cdot n!}$.

We are only interested in valid pathways: these are trajectories where all customers can successfully buy a ticket. A pathway is valid if it never intersects or crosses the horizontal line $y = -1$.

This is because a \$10 person can only be served if there was a \$5 person in line before them. For example, consider a scenario where the first person in line has a \$5 bill and the second person has a \$10 bill. In such a situation, the initial segment of the corresponding path would look like this:

The sequence of points $(0,0) \to (1,1) \to (2,0)$ represents a movement pattern.

The application of the reflection principle can be used to enumerate the quantity of paths that are invalid. It's important to note that all paths, whether valid or invalid, initiate from $A = (0,0)$ and conclude at $B = (2n,0)$. Now, let's focus on an invalid path. By virtue of being invalid, there exists a specific point (let's denote it as point C) along this path where it intersects the line $y = -1$ (since otherwise, it would be a valid path). Thus, we can express C as $(x,-1)$, where $x > 0$ and $x < 2n$.

Next, we construct the mirrored path (as depicted in the illustration): this mirrored path coincides with the original path from A to C and then becomes reflected across the line $y = -1$ between points C and B. Given that the initial path terminates at B, this newly mirrored path concludes at $\tilde{B} = (2n,-2)$. In summary, the mirrored path traverses from A to $\tilde{B}$.

Take note that each inadmissible path establishes a one-to-one correspondence with a reflected path. Consequently, the count of inadmissible paths mirrors that of the reflected inadmissible paths.

All inadmissible paths initiate from $A = (0,0)$ and conclude at $\tilde{B} = (2n,-2)$. This scenario is akin to our initial problem with $n-1$ \$5 people and $n+1$ \$10 people. Thus, the tally of inadmissible paths equates to

$$N_{\text{invalid}} = \binom{2n}{n+1} = \frac{(2n)!}{(n+1)!(n-1)!}.$$

Consequently, the quantity of valid paths becomes

$$N_{\text{valid}} = N_{\text{total}} - N_{\text{invalid}}.$$

Ultimately, the probability of a valid path is given by

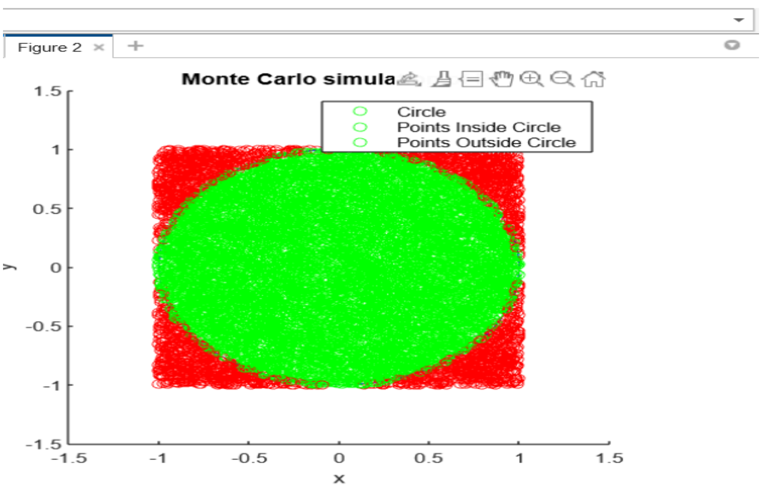
$$P = \frac{N_{\text{valid}}}{N_{\text{total}}} = 1 - \frac{n! \cdot n!}{(n+1)! \cdot (n-1)!} = 1 - \frac{n}{n+1}.$$

$$P = \frac{1}{n+1}.$$

Now for $n = \frac{k\pi}{2}$, we will estimate the answer using Monte Carlo simulation. Last digit of my roll no. is 8, for $k = 8, n = 4\pi$. Then

$$P = \frac{1}{1+4\pi}.$$

Now this is the python code,



Graph of Monte Carlo simulation

Monte_c.m *

/MATLAB Drive/Monte_c.m

```matlab
I = 10;
circle_points = 0;
square_points = 0;

for i = 1:(I^2)
    % Random values of x and y
    % Range of x and y is -1 to 1
    rnd_x = -1 + 2 * rand();
    rnd_y = -1 + 2 * rand();

    % Distance between (x, y) from the origin
    r = rnd_x^2 + rnd_y^2;

    % Checking that (x, y) lies inside the circle
    if r <= 1
        circle_points = circle_points + 1;
    end

    square_points = square_points + 1;
    % pi = 4 * (no. of points generated inside the circle/ (no. of points generated inside the square)
    pi = 1/(1+16 * circle_points / square_points);
end
pi
% Plotting the points inside and outside the circle
figure;
```

MATLAB code for Monte Carlo simulation

Monte_c.m *

/MATLAB Drive/Monte_c.m

```matlab
% Plotting the points inside and outside the circle
figure;
hold on;
rectangle('Position', [-1, -1, 2, 2], 'Curvature', [1, 1], 'EdgeColor', 'b');
axis equal;
xlim([-1.5, 1.5]);
ylim([-1.5, 1.5]);
for i = 1:square_points
    rand_x = -1 + 2 * rand();
    rand_y = -1 + 2 * rand();

    origin_dist = rand_x^2 + rand_y^2;

    if origin_dist <= 1
        plot(rand_x, rand_y, 'go');
    else
        plot(rand_x, rand_y, 'ro');
    end
end
title('Monte Carlo simulation of Pi');
xlabel('x');
ylabel('y');
legend('Circle', 'Points Inside Circle', 'Points Outside Circle');
hold off;
```

MATLAB code for plotting Monte Carlo simulation

Q.2 A series of cups of equal capacity have been filled with water and arranged one below another. Pour into the first cup a quantity of wine equal to the capacity of the cup at a constant rate and let the overflow in each cup, go into the cup just below. Assuming that complete mixing of wine and water takes place instantaneously. Find the amount of wine in each cup at any time t and at the end of the process at time T. Also solve when the rate of flow is not constant. For both cases plot the amount of wine in n^{th} cup for different values of capacity, where n is the last non-zero digit of your roll number. Again for both cases assuming a fixed value (2^n unit) of capacity, plot the amount of wine in $(n+10)$ different cups.

Ans. Let q denote the capacity of each cup, so that the rate of flow is q/T. Let x_n be the amount of wine in the nth cup and x_{n-1} that in the $(n-1)$th cup. Then the rate of flow of wine into the nth cup is $\frac{x_{n-1}}{T}$ and out of it is $\frac{x_n}{T}$. This gives

$$\frac{dx_n}{dt} = \frac{x_{n-1}}{T} - \frac{x_n}{T} \tag{1}$$

Any equation in the sequence can be solved if the one before it has been solved. Thus,

$$\frac{dx_1}{dt} = \frac{q}{T} - \frac{x_1}{T} \tag{2}$$

which gives

$$x_1 = q\left(1 - \frac{1}{e^{t/T}}\right) \tag{3}$$

For $n = 2$ we have,

$$\begin{aligned}\frac{dx_2}{dt} &= \frac{x_1}{T} - \frac{x_2}{T}\\ \frac{dx_2}{dt} &= \frac{q\left(1 - \frac{1}{e^{t/T}}\right)}{T} - \frac{x_2}{T}\end{aligned}$$

which gives

$$x_2 = q\left[1 - \frac{1 + (1/1!)(t/T)}{e^{t/T}}\right]. \tag{4}$$

For the final amounts in the successive cups one has

$$x_n = q\left(1 - \frac{\left(1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{(n-1)!}\right)}{e}\right) \tag{5}$$

$k \to \infty$, $x_4 \to 0$.

Note that the rate of flow is not essential. If x, the amount of wine poured into the first cup, is the independent variable, then

$$\begin{aligned}\frac{dx_1}{dx} + \frac{x_1}{q} &= 1\\ \frac{dx_2}{dx} + \frac{x_2}{q} &= \frac{x_1}{q}\\ \vdots \quad \vdots \quad &\vdots\\ \frac{dx_n}{dx} + \frac{x_n}{q} &= \frac{x_{n-1}}{q}\end{aligned}$$

and

$$x_n = q\left[1 - \left(1 + \frac{1}{1!}\left(\frac{x}{q}\right) + \frac{1}{2!}\left(\frac{x}{q}\right)^2 + \frac{1}{3!}\left(\frac{x}{q}\right)^3 + \ldots + \frac{1}{(n-1)!}\left(\frac{x}{q}\right)^{n-1}\right)e^{-x/q}\right].$$

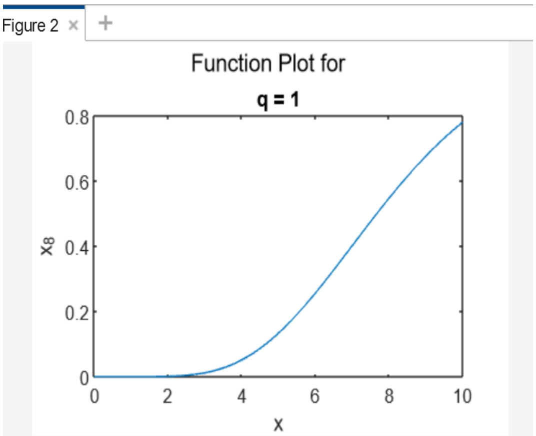
This final result is obtained by taking $x = q$.

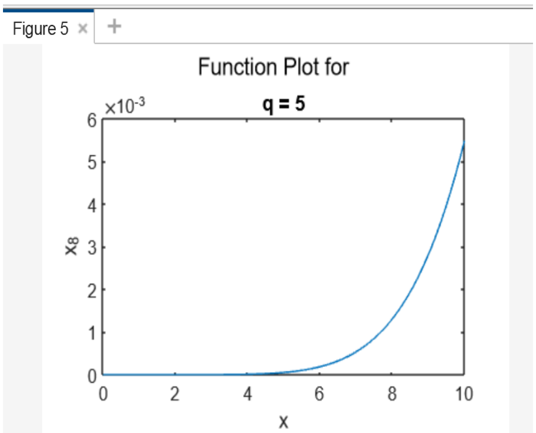
NLA_1.m Model_A1.m Model_A2.m

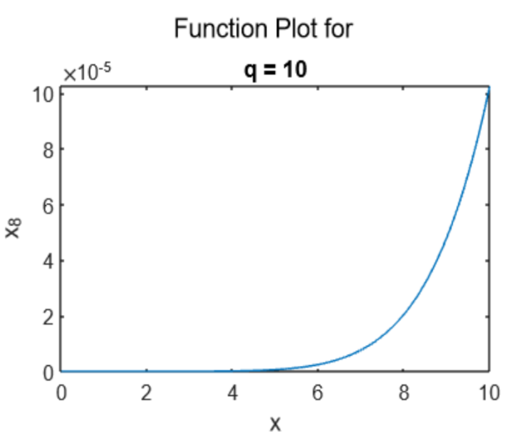
MATLAB Drive/Model_A1.m

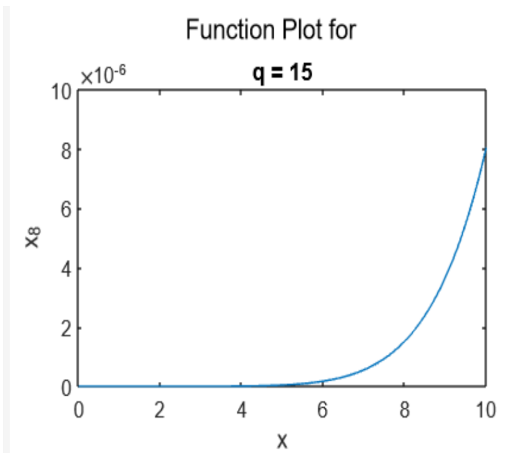
```matlab
% Range of x
x = linspace(0, 10, 100);
q_values = [15];

figure;

for q_idx = 1:length(q_values)
    q = q_values(q_idx);

    n = 8;
    func_values = zeros(size(x));
    for i = 1:length(x)
        term_sum = 0;
        for j = 0:(n-1)
            term = (x(i)/q)^j / factorial(j);
            term_sum = term_sum + term;
        end
        func_values(i) = q * (1 - term_sum * exp(-x(i)/q));
    end

    % Plot the function
    subplot(length(q_values), 1, q_idx);
    plot(x, func_values);
    title(['q = ', num2str(q)]);
    xlabel('x');
    ylabel(['x_', num2str(n)]);
end
sgtitle('Function Plot for ');
```

MATLAB code to plot amount of wine in $8th$ cup for different values of capacity(q).







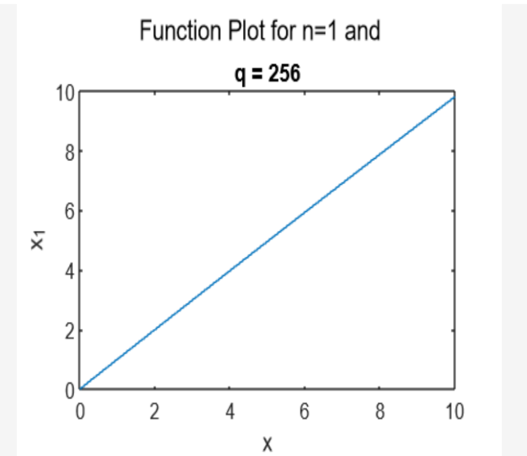


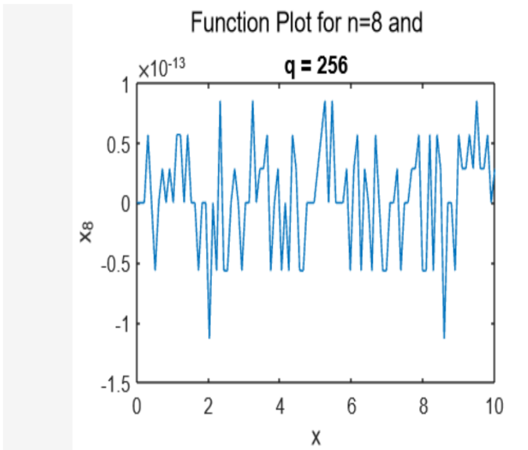
NLA_1.m × Model_A1.m × Model_A2.m × +

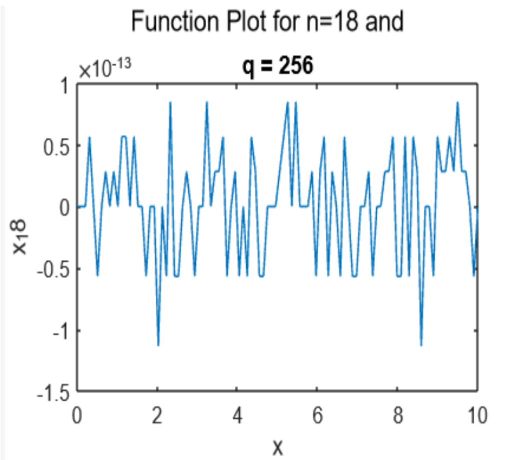
/MATLAB Drive/Model_A2.m

```matlab
% Range of x
x = linspace(0, 10, 100);
q_value = 256;

figure;

for q_idx = 1:length(q_value)
    q = q_value(q_idx);

    n = 18;
    func_values = zeros(size(x));
    for i = 1:length(x)
        term_sum = 0;
        for j = 0:(n-1)
            term = (x(i)/q)^j / factorial(j);
            term_sum = term_sum + term;
        end
        func_values(i) = q * (1 - term_sum * exp(-x(i)/q));
    end

    % Plot the function
    subplot(length(q_value), 1, q_idx);
    plot(x, func_values);
    title(['q = ', num2str(q)]);
    xlabel('x');
    ylabel(['x_', num2str(n)]);
end
sgtitle('Function Plot for n=18 and');
```

MATLAB code to plot amount of wine in different cup for capacity($q = 2^8 = 256$).







Now we will solve it when the rate of flow is not constant.

Let's denote the rate of flow of wine into the first cup as a function of time, $r(t)$. The rate of flow into the nth cup will be proportional to the amount of wine in the $(n-1)$th cup at that time. Mathematically, this can be expressed as:

$$\frac{dx_n}{dt} = r(t) \cdot x_{n-1}(t) - \frac{x_n(t)}{T} \quad (1)$$

where $x_n(t)$ is the amount of wine in the kth cup at time t.

We need to specify the function $r(t)$ to fully solve the problem.

Let's consider a simple case where $r(t)$ is linear with time:

$$r(t) = a + bt$$

where a and b are constants.

Starting with the first cup $(n = 1)$:

$$\frac{dx_1}{dt} = (a + bt) \cdot x_0 - \frac{x_1}{T}$$
$$\frac{dx_1}{dt} = (a + bt) \cdot q - \frac{x_1}{T}$$

This is a first-order linear ordinary differential equation. We can separate variables and integrate:

$$\frac{1}{q - x_1} dx_1 = (a + bt)\, dt$$

Integrating both sides:

$$-\ln|q - x_1| = at + \frac{b}{2}t^2 + C_1$$

Solving for x_1:

$$q - x_1 = e^{-at-\frac{b}{2}t^2-C_1}$$
$$x_1 = q - e^{-at-\frac{b}{2}t^2-C_1}$$

Now, for the second cup $(n = 2)$:

$$\frac{dx_2}{dt} = (a + bt) \cdot x_1 - \frac{x_2}{T}$$

Substitute the expression for x_1:

$$\frac{dx_2}{dt} = (a + bt) \cdot \left(q - e^{-at-\frac{b}{2}t^2-C_1}\right) - \frac{x_2}{T}$$

This is another first-order linear ordinary differential equation. We'll follow the same steps to solve it:

$$\frac{dx_2}{dt} = (a + bt) \cdot q - (a + bt) \cdot e^{-at-\frac{b}{2}t^2-C_1} - \frac{x_2}{T}$$

Integrating both sides:

$$-\ln|q - x_2| = q(at + \frac{b}{2}t^2) - \frac{a}{2}t^2 - C_1 t + C_2$$

Solving for x_2:

$$q - x_2 = e^{q(at+\frac{b}{2}t^2)-\frac{a}{2}t^2-C_1t+C_2}$$

$$x_2 = q - e^{q(at+\frac{b}{2}t^2)-\frac{a}{2}t^2-C_1t+C_2}$$

For the third cup ($n = 3$):

$$\frac{dx_3}{dt} = (a+bt)\cdot\left(q - e^{q(at+\frac{b}{2}t^2)-\frac{a}{2}t^2-C_1t+C_2}\right) - \frac{x_3}{T}$$

This is another first-order linear ordinary differential equation. We'll follow the same steps to solve it:

$$\frac{dx_3}{dt} = (a+bt)\cdot q - (a+bt)\cdot e^{q(at+\frac{b}{2}t^2)-\frac{a}{2}t^2-C_1t+C_2} - \frac{x_3}{T}$$

Integrating both sides:

$$-\ln|q - x_3| = q(at + \frac{b}{2}t^2) - \frac{a}{2}t^2 - C_1t + C_2 - \frac{a}{2T}t^2 + C_3$$

Solving for x_3:

$$q - x_3 = e^{q(at+\frac{b}{2}t^2)-\frac{a}{2}t^2-C_1t+C_2-\frac{a}{2T}t^2+C_3}$$

$$x_3 = q - e^{q(at+\frac{b}{2}t^2)-\frac{a}{2}t^2-C_1t+C_2-\frac{a}{2T}t^2+C_3}$$

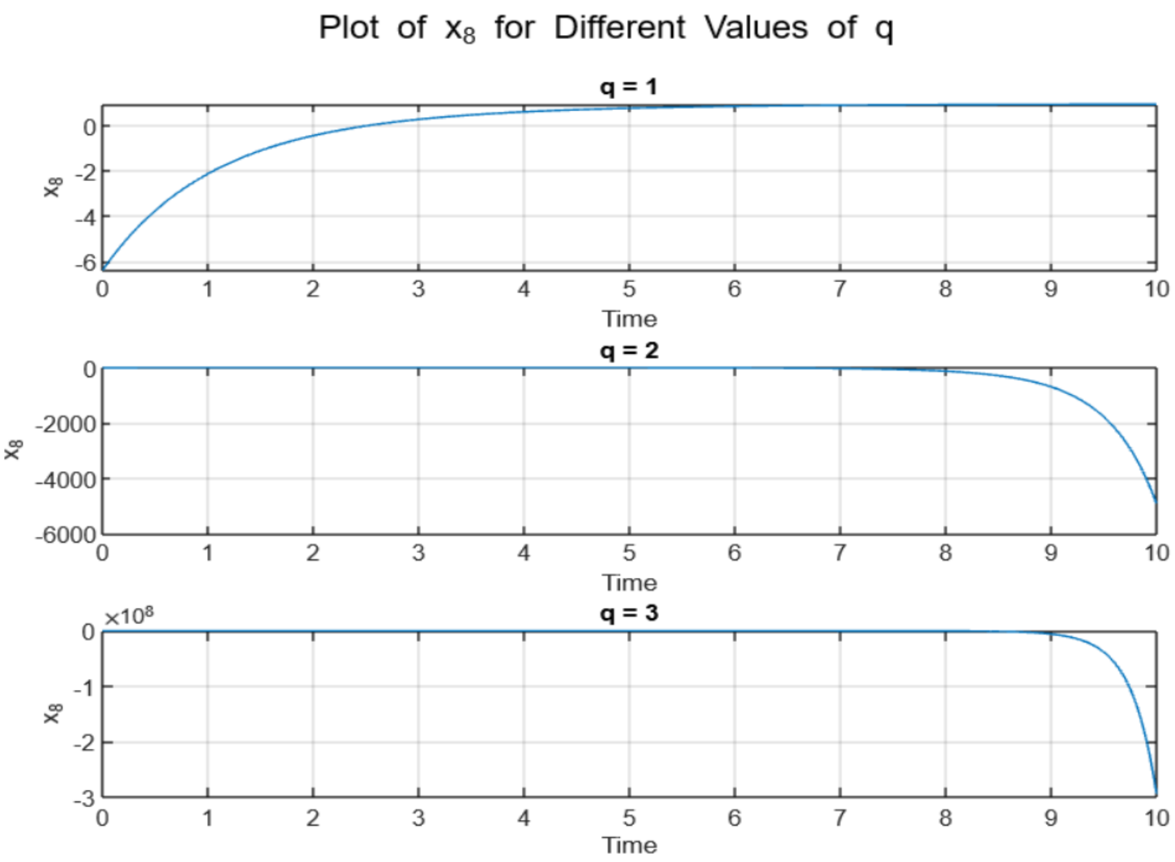
You can apply this procedure iteratively for each subsequent cup, plugging in the expression for x_{n-1} into the equation for x_n and solving the resulting differential equation. Each solution will involve its own set of constants C_n.

NLA_1.m × Model_A1.m × Model_A2.m × Model_A3.m * ×

/MATLAB Drive/Model_A3.m

```
a = 0.1;
b = 0.2;
T = 10;
t = linspace(0, 10, 1000);

% Different values of q
q_values = [1, 2, 3];

figure;
for i = 1:length(q_values)
    q = q_values(i);
    x_8 = q - exp(q*(a*t + (b/2)*t.^2) - (a/2)*t.^2 - t + 1 - (a/(2*T))*t.^2 + 1);

    % Plot the function
    subplot(length(q_values), 1, i);
    plot(t, x_8);
    title(['q = ' num2str(q)]);
    xlabel('Time');
    ylabel('x_8');
    grid on;
end

% Adjust layout
sgtitle('Plot of x_8 for Different Values of q');
```

MATLAB code to plot amount of wine in $8th$ cup for different values of capacity(q).

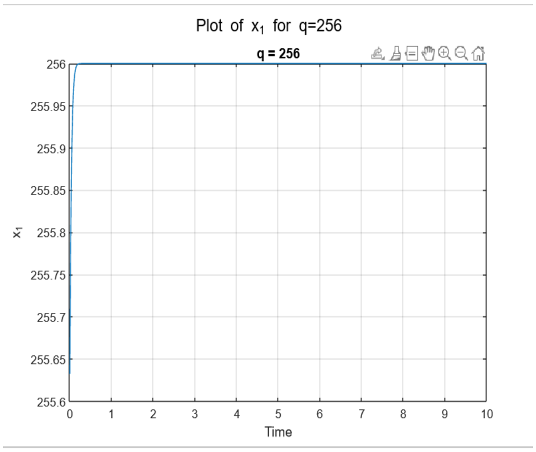


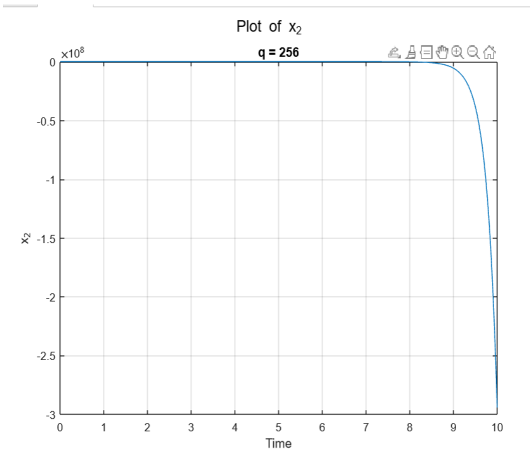
NLA_1.m × | Model_A1.m × | Model_A2.m × | Model_A3.m × | Model_A4.m × | +

/MATLAB Drive/Model_A4.m

```
a = 0.1;
b = 0.2;
T = 10;
t = linspace(0, 10, 1000);

% Different values of q
q_value = [256];

figure;
for i = 1:length(q_value)
    q = q_value(i);
    x_2 = q - exp(q*(a*t + (b/2)*t.^2) - (a/2)*t.^2 - t + 1);

    % Plot the function
    subplot(length(q_value), 1, i);
    plot(t, x_8);
    title(['q = ' num2str(q)]);
    xlabel('Time');
    ylabel('x_2');
    grid on;
end

% Adjust layout
sgtitle('Plot of x_2');

```

MATLAB code to plot amount of wine in different cup for capacity$(q = 2^8 = 256)$.





Q.3 A grocery seller sells rice at a profit of Rs. 2.00 per kg and loses Rs. 5.00 for each unsold kg of rice. Assuming that the demand is unknown but can be estimated from previous data as being given by the probability density $f(y)$, with a minimum sale of y_0 kg of rice. Calculate the amount of rice the grocery seller should store in order to maximize the expected profit.

(a) Assume $f(y) = 2y + 1$. Now compute (Analytically and Numerically) y_0.

(b) Assume $f(y) = ay + b$, where $a - b = 1$. For $y_0 = 1$ compute (Analytically and Numerically) the amount of rice the grocery seller should store in order to maximize the expected profit.

Ans. To maximize the expected profit for the grocery seller, we need to find the optimal amount of rice to store. Let's break down the problem step by step:

Let:

$$y_0 = \text{minimum sale in kg},$$
$$f(y) = \text{probability density function for the demand of rice in kg},$$
$$p = \text{profit per kg (Rs. 2.00)},$$
$$c = \text{loss per unsold kg (Rs. 5.00)}.$$

Case 1: The amount of rice the seller stores (x) is less than the demand (y).
Case 2: The demand (y) is less than the amount of rice the seller stores (x).
We will calculate the expected profit for each case and then combine them to find the overall expected profit.

For Case 1 ($x < y$): In this case, the seller will sell all x kg of rice at a profit p and incur a loss of $(y - x)$ kg of rice at a cost c. The expected profit from this case can be calculated as:

$$E_1 = p \cdot \int_{y_0}^{x} y f(y)\, dy$$

For Case 2 ($x > y$): In this case, the seller will sell all y kg of rice at a profit p and incur a loss of $(x - y)$ kg of rice at a cost c. The expected profit from this case can be calculated as:

$$E_2 = \int_{0}^{y_0} (py - cx + cy) f(y)\, dy$$

Overall Expected Profit:

$$E = E_1 + E_2$$

$$E = p \cdot \int_{y_0}^{x} y f(y)\, dy + \int_{0}^{y_0} (py - cx + cy) f(y)\, dy$$

To maximize function E, Now, differentiate E with respect to x:

$$\frac{dE}{dx} = \frac{d}{dx}\left(p \cdot \int_{y_0}^{x} y f(y)\, dy + \int_{0}^{y_0} (py - cx + cy) f(y)\, dy\right)$$

Using the Leibniz rule for differentiating under the integral sign, we have:

$$\frac{dE}{dx} = pxf(x) + \int_{0}^{y_0} (-c) f(y)\, dy$$

Now, we want to solve for $\frac{dE}{dx} = 0$:

$$pxf(x) + \int_{0}^{y_0} (-c) f(y)\, dy = 0$$

$$\implies \int_{0}^{y_0} f(y)\, dy = \frac{p}{c} x f(x).$$

(a.) Since $f(y) = 2y+1$ is probability density function then, we start by evaluating the integral:

$$\int_{0}^{y_0} (2y+1)\, dy = 1$$

Integrating with respect to y gives:

$$\left[y^2 + y\right]_0^{y_0} = y_0^2 + y_0 - (0^2 + 0) = y_0^2 + y_0$$

Now we have the equation:

$$y_0^2 + y_0 = 1$$

$$\implies y_0^2 + y_0 - 1 = 0$$

Now we can solve for y_0 using the quadratic formula:

$$y_0 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

For our equation, $a = 1$, $b = 1$, and $c = -1$. Putting these values in:

$$y_0 = \frac{-1 \pm \sqrt{1^2 - 4(1)(-1)}}{2(1)}$$

$$y_0 = \frac{-1 \pm \sqrt{5}}{2}$$

For

$$y_0 = \frac{-1 + \sqrt{5}}{2}$$

So, the value of y_0 is approximately 0.618.
Now using Newton Raphson method value of y_0 is

Model_A4.m × Model_A3.m × Model_A2.m × Model_A1.m × NLA_1.m × Model_A5.m * × +

/MATLAB Drive/Model_A5.m

```matlab
% Define the equation: f(y) = y^2 + y - 1
f = @(y) y^2 + y - 1;
% Define the derivative of the equation: f'(y) = 2*y + 1
df = @(y) 2*y + 1;
% Initial guess
y0_guess = 0;
% Set the tolerance and maximum number of iterations
tolerance = 1e-6;
max_iterations = 6;

% Initialize variables
y0 = y0_guess;
iterations = 0;
error = Inf;
% Newton-Raphson iteration loop
while error > tolerance && iterations < max_iterations
    y0_new = y0 - f(y0) / df(y0);
    error = abs(y0_new - y0);
    y0 = y0_new;
    iterations = iterations + 1;
end

% Display the result
if iterations < max_iterations
    fprintf('Solution found after %d iterations:\n', iterations);
    fprintf('y_0 = %.6f\n', y0);
else
    fprintf('Solution not found after %d iterations.\n', max_iterations);
end

```

MATLAB code of Newton Raphson method to find value of y_0.

b. Given the probability density function $f(y) = ay + b$ and the condition $a - b = 1$, we want to find the values of a and b such that $\int_0^{y_0} f(y)\,dy = 1$ for $y_0 = 1$.

The integral $\int_0^{y_0} f(y)\,dy$ can be calculated as follows:

$$\int_0^{y_0} f(y)\,dy = \int_0^1 (ay + b)\,dy = \left[\frac{a}{2}y^2 + by\right]_0^1 = \frac{a}{2} + b - (0) = \frac{a}{2} + b.$$

Since we want this integral to be equal to 1, we set up the equation:

$$\frac{a}{2} + b = 1.$$

We are also given the condition that $a - b = 1$, so we can rewrite this equation as:

$$\frac{a}{2} + (a - 1) = 1.$$

Solving for a:

$$\frac{a}{2} + a - 1 = 1,$$

$$\frac{3}{2}a = 2,$$

$$a = \frac{4}{3}.$$

Using the condition $a - b = 1$:

$$\frac{4}{3} - b = 1,$$

$$b = \frac{1}{3}.$$

So, the values of a and b that satisfy both conditions are $a = \frac{4}{3}$ and $b = \frac{1}{3}$.
To maximize x, given:

$$f(y) = \frac{4}{3}y + 1, \quad y_0 = 1, \quad \int_0^{y_0} f(y)\,dy = cp \cdot xf(x).$$

Compute the definite integral of $f(y)$ from 0 to y_0:

$$\int_0^{y_0} f(y)\,dy = \int_0^1 \left(\frac{4}{3}y + 1\right)\,dy.$$

We'll split the integral into two parts:

$$\frac{1}{3}\int_0^1 (4y+1)\,dy = \frac{1}{3}\left[\int_0^1 4y\,dy + \int_0^1 1\,dy\right].$$

Simplify and evaluate the integrals:

$$\frac{1}{3}\left[2y^2\Big|_0^1 + y\Big|_0^1\right] = \frac{1}{3}\left[(2 \cdot 1^2) - (2 \cdot 0^2) + (1 - 0)\right] = \frac{1}{3} \cdot 3 = 1.$$

So, the left-hand side becomes:

$$\int_0^{y_0} f(y)\,dy = 1.$$

Substitute the result back into the equation:

$$1 = cp \cdot xf(x).$$

Solve for x:

$$x = \frac{1}{cpf(x)}.$$

Since $f(y) = \frac{4}{3}y + 1$, we have:

$$x = \frac{1}{cp(\frac{4}{3}x + 1)}.$$

Since $p = 2, c = 5$ then we have,

$$x = \frac{1}{10(\frac{4}{3}x + 1)}.$$

$$\implies 40x^2 + 30x - 3 = 0.$$

Now we can solve for x using the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

For our equation, $a = 40$, $b = 30$, and $c = -3$. Putting these values in:

$$x = \frac{-30 \pm \sqrt{30^2 - 4(40)(-3)}}{2(40)}$$

$$\implies x = \frac{-30 \pm \sqrt{780}}{80}$$

$$\implies x = -0.83935, 0.089354 \text{ Kg}.$$

/MATLAB Drive/Model_A6.m

```
% Define the equation: f(x)
f = @(x) 40*x^2 + 30*x - 3;
% Define the derivative of the equation: f'(x)
df = @(x) 80*x + 30;
% Initial guess
x_guess = 0;
% Set the tolerance and maximum number of iterations
tolerance = 1e-6;
max_iterations = 6;

% Initialize variables
x = x_guess;
iterations = 0;
error = Inf;
% Newton-Raphson iteration loop
while error > tolerance && iterations < max_iterations
    x_new = x - f(x) / df(x);
    error = abs(x_new - x);
    x = x_new;
    iterations = iterations + 1;
end

% Display the result
if iterations < max_iterations
    fprintf('Solution found after %d iterations:\n', iterations);
    fprintf('y_0 = %.6f\n', x);
else
    fprintf('Solution not found after %d iterations.\n', max_iterations);
end
```

MATLAB code of Newton Raphson method to find value of x.