**Implementing the SIR Model for Epidemiological Analysis in MATLAB**

**Abstract**

In this exercise, I revisited the implementation of Euler's method in MATLAB to model population dynamics in the context of susceptible (S), infected (I), and recovered (R) groups. I corrected critical errors in my earlier code, emphasizing accurate numerical solutions and ensuring stability over time. By visualizing these population trends, I explored the effects of transmission rates, recovery times, and step sizes on the behavior of the system. Through small incremental changes, I aimed to refine the simulation for better accuracy and visual clarity, demonstrating the power of numerical methods and computational tools in understanding epidemiological models.

**Essay**

Numerical methods are indispensable for solving real-world problems that lack closed-form solutions. Euler's method, a fundamental tool for solving ordinary differential equations, offers an intuitive approach to stepping through changes in a dynamic system. When applied to epidemiological models like the SIR framework, it provides a powerful means to simulate population dynamics over time.

**Correcting Errors in Euler's Method**

Initially, I realized my implementation of Euler's method was flawed, as I had neglected to multiply the rate of change (dydx\frac{dy}{dx}dxdy​) by the step size (Δx\Delta xΔx). This oversight led to unrealistic results, such as negative susceptible populations. To address this, I revisited the mathematical foundation of Euler's method, ensuring that every incremental change was scaled appropriately. I also capped negative values to maintain logical consistency in the model.

**Impact of Step Size**

Step size plays a pivotal role in numerical stability and accuracy. Large steps yielded "chunky" results, while smaller steps smoothed the curves and improved precision. By leveraging MATLAB's computational power, I was able to use extremely small step sizes to refine the simulation without performance concerns.

**Visualizing and Refining Results**

Visualization is key to understanding numerical simulations. By plotting S, I, and R populations over time, I observed the interplay between infection spikes and recovery rates. I added gridlines, legends, and axis labels to enhance interpretability. Adjusting parameters like transmission rate and recovery time provided additional insights, demonstrating how small changes in these variables can dramatically shift population trends.

**MATLAB Code**

matlab

Copy code

% Corrected Euler's Method for SIR Model

% This code models susceptible (S), infected (I), and recovered (R) populations

% over time using Euler's method, ensuring accurate and realistic results.

% Clear workspace

% I always clear my workspace to avoid conflicts and ensure the environment is clean.

clear; clc;

% Parameters

% I chose these values to explore a realistic epidemiological scenario.

N = 1e6; % Total population

I0 = 1; % Initial infected

S0 = N - I0; % Initial susceptible

R0 = 0; % Initial recovered

beta = 2; % Transmission rate (per person per week)

gamma = 0.5; % Recovery rate (per week)

dt = 0.001; % Time step (weeks)

T\_max = 15; % Maximum simulation time (weeks)

% Time vector

% I created a time vector to iterate through weeks for the simulation.

t = 0:dt:T\_max;

% Initialize populations

% I preallocated arrays for efficiency and set initial values.

S = zeros(size(t));

I = zeros(size(t));

R = zeros(size(t));

S(1) = S0;

I(1) = I0;

R(1) = R0;

% Euler's method loop

% Here, I iteratively calculated changes in S, I, and R using differential equations.

for n = 1:length(t) - 1

dS = -beta \* S(n) \* I(n) / N; % Rate of change of susceptible

dI = beta \* S(n) \* I(n) / N - gamma \* I(n); % Rate of change of infected

dR = gamma \* I(n); % Rate of change of recovered

% Update populations

% I scaled the rates by the step size to calculate the actual change.

S(n + 1) = S(n) + dS \* dt;

I(n + 1) = I(n) + dI \* dt;

R(n + 1) = R(n) + dR \* dt;

% Prevent negative populations

% I added this condition to ensure logical consistency.

if S(n + 1) < 0, S(n + 1) = 0; end

if I(n + 1) < 0, I(n + 1) = 0; end

if R(n + 1) < 0, R(n + 1) = 0; end

end

% Plot results

% Visualization is critical, so I carefully plotted each population with labels.

figure;

plot(t, S, '-b', 'LineWidth', 2); hold on;

plot(t, I, '-r', 'LineWidth', 2);

plot(t, R, '-g', 'LineWidth', 2);

xlabel('Time (weeks)');

ylabel('Number of People');

title('SIR Model: Population Dynamics');

legend('Susceptible', 'Infected', 'Recovered', 'Location', 'Northeast');

grid on;

grid minor;

% Reflections

% By iterating through this process, I improved both my understanding of the SIR model

% and the practical application of Euler's method. MATLAB's flexibility allowed me to

% explore various parameter configurations efficiently, providing valuable insights

% into the dynamics of disease spread and recovery.