%% COVID-19 Model Fitting for Nigeria Data

% This script fits the COVID-19 compartmental model to Nigeria data

% and estimates the model parameters

clear;

close all;

%% Load Nigeria COVID-19 data (Jul 9 - Aug 7, 2021)

% Define dates and corresponding case numbers - Fixed with explicit format

dates = datetime({'Jul-09-2021', 'Jul-10-2021', 'Jul-11-2021', 'Jul-12-2021', 'Jul-13-2021', ...

'Jul-14-2021', 'Jul-15-2021', 'Jul-16-2021', 'Jul-17-2021', 'Jul-18-2021', ...

'Jul-19-2021', 'Jul-20-2021', 'Jul-21-2021', 'Jul-22-2021', 'Jul-23-2021', ...

'Jul-24-2021', 'Jul-25-2021', 'Jul-26-2021', 'Jul-27-2021', 'Jul-28-2021', ...

'Jul-29-2021', 'Jul-30-2021', 'Jul-31-2021', 'Aug-01-2021', 'Aug-02-2021', ...

'Aug-03-2021', 'Aug-04-2021', 'Aug-05-2021', 'Aug-06-2021', 'Aug-07-2021'}, ...

'InputFormat', 'MMM-dd-yyyy');

cases = [11713, 11515, 11421, 10357, 10363, 10243, 10237, 10126, 9174, 9231, ...

9170, 9202, 9139, 9227, 7700, 7594, 7518, 7520, 7578, 7626, ...

7599, 7625, 7669, 7734, 7782, 7821, 7840, 7986, 7949, 7929];

% Convert to days since first data point

days = 0:length(cases)-1;

%% Define parameter bounds for optimization

% Parameter ordering:

% [Lambda, beta\_1, beta\_2, chi\_1, chi\_2, psi\_1, psi\_2, eta, mu, mu\_prime,

% lambda, phi, sigma\_1, sigma\_2, delta\_1, delta\_2, delta\_3, gamma, omega]

% Lower bounds

lb = [50, % Lambda (recruitment rate)

0.1, % beta\_1 (infection rate from infected)

0.1, % beta\_2 (infection rate from hospitalized)

0.01, % chi\_1 (vaccination rate)

0.01, % chi\_2 (waning immunity rate)

0.01, % psi\_1 (quarantined to hospitalized rate)

0.01, % psi\_2 (quarantined to susceptible rate)

0.01, % eta (immunity loss rate)

0.001, % mu (natural death rate)

0.001, % mu\_prime (death rate for treated)

0.05, % lambda (exposed to infected rate)

0.05, % phi (exposed to quarantined rate)

0.05, % sigma\_1 (infected to hospitalized rate)

0.05, % sigma\_2 (infected to treated rate)

0.01, % delta\_1 (disease-induced death rate for infected)

0.01, % delta\_2 (disease-induced death rate for treated)

0.01, % delta\_3 (disease-induced death rate for hospitalized)

0.05, % gamma (hospitalized to treated rate)

0.05]; % omega (recovery rate)

% Upper bounds

ub = [500, % Lambda

1.0, % beta\_1

1.0, % beta\_2

0.5, % chi\_1

0.5, % chi\_2

0.5, % psi\_1

0.5, % psi\_2

0.5, % eta

0.05, % mu

0.1, % mu\_prime

1.0, % lambda

1.0, % phi

1.0, % sigma\_1

1.0, % sigma\_2

0.5, % delta\_1

0.5, % delta\_2

0.5, % delta\_3

1.0, % gamma

1.0]; % omega

% Initial guess for parameters (midpoint between bounds)

x0 = (lb + ub) / 2;

%% Define the function to compute the MSE between model output and data

function mse = objectiveFunction(params, days, observed\_cases)

% Extract parameters

Lambda = params(1);

beta\_1 = params(2);

beta\_2 = params(3);

chi\_1 = params(4);

chi\_2 = params(5);

psi\_1 = params(6);

psi\_2 = params(7);

eta = params(8);

mu = params(9);

mu\_prime = params(10);

lambda\_param = params(11);

phi = params(12);

sigma\_1 = params(13);

sigma\_2 = params(14);

delta\_1 = params(15);

delta\_2 = params(16);

delta\_3 = params(17);

gamma = params(18);

omega = params(19);

% Create parameter structure for the model

model\_params = struct();

model\_params.Lambda = Lambda;

model\_params.beta\_1 = beta\_1;

model\_params.beta\_2 = beta\_2;

model\_params.chi\_1 = chi\_1;

model\_params.chi\_2 = chi\_2;

model\_params.psi\_1 = psi\_1;

model\_params.psi\_2 = psi\_2;

model\_params.eta = eta;

model\_params.mu = mu;

model\_params.mu\_prime = mu\_prime;

model\_params.lambda = lambda\_param;

model\_params.phi = phi;

model\_params.sigma\_1 = sigma\_1;

model\_params.sigma\_2 = sigma\_2;

model\_params.delta\_1 = delta\_1;

model\_params.delta\_2 = delta\_2;

model\_params.delta\_3 = delta\_3;

model\_params.gamma = gamma;

model\_params.omega = omega;

% Initial population estimates

% Assume total population of Nigeria ~200 million

N\_total = 200000000;

% Initial breakdown based on observed cases

% Assume initial infected is the first data point

I\_initial = observed\_cases(1);

% Make some reasonable assumptions for other compartments

E\_initial = I\_initial \* 2; % Assume twice as many exposed as infected

Q\_initial = I\_initial \* 0.5; % Assume half as many quarantined as infected

J\_initial = I\_initial \* 0.3; % Assume 30% of infected are hospitalized

T\_initial = I\_initial \* 0.2; % Assume 20% of infected are under treatment

R\_initial = I\_initial \* 5; % Assume 5 times as many recovered as infected

V\_initial = N\_total \* 0.1; % Assume 10% of population is vaccinated

% Calculate susceptible as remainder

S\_initial = N\_total - E\_initial - I\_initial - Q\_initial - J\_initial - T\_initial - V\_initial - R\_initial;

% Initial conditions [S, E, I, Q, J, T, V, R]

initial\_conditions = [S\_initial, E\_initial, I\_initial, Q\_initial, J\_initial, T\_initial, V\_initial, R\_initial];

% Time span for simulation

tspan = [0 max(days)];

% Solve the ODE system

options = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);

[t, y] = ode45(@(t, y) COVID19Model(t, y, model\_params), tspan, initial\_conditions, options);

% Interpolate model output to match observed data days

I\_model = interp1(t, y(:, 3), days);

% Calculate mean squared error

mse = mean((I\_model - observed\_cases).^2);

end

%% Optimize model parameters using fmincon

options = optimoptions('fmincon', 'Display', 'iter', 'MaxFunctionEvaluations', 1000, ...

'MaxIterations', 200, 'Algorithm', 'sqp');

objective = @(params) objectiveFunction(params, days, cases);

disp('Starting parameter optimization...');

tic;

[opt\_params, fval] = fmincon(objective, x0, [], [], [], [], lb, ub, [], options);

toc;

disp('Optimization complete.');

%% Extract optimized parameters

params = struct();

params.Lambda = opt\_params(1);

params.beta\_1 = opt\_params(2);

params.beta\_2 = opt\_params(3);

params.chi\_1 = opt\_params(4);

params.chi\_2 = opt\_params(5);

params.psi\_1 = opt\_params(6);

params.psi\_2 = opt\_params(7);

params.eta = opt\_params(8);

params.mu = opt\_params(9);

params.mu\_prime = opt\_params(10);

params.lambda = opt\_params(11);

params.phi = opt\_params(12);

params.sigma\_1 = opt\_params(13);

params.sigma\_2 = opt\_params(14);

params.delta\_1 = opt\_params(15);

params.delta\_2 = opt\_params(16);

params.delta\_3 = opt\_params(17);

params.gamma = opt\_params(18);

params.omega = opt\_params(19);

% Display optimized parameters

disp('Optimized Parameters:');

disp(params);

%% Simulate with optimized parameters

% Initial conditions as in the objective function

N\_total = 200000000;

I\_initial = cases(1);

E\_initial = I\_initial \* 2;

Q\_initial = I\_initial \* 0.5;

J\_initial = I\_initial \* 0.3;

T\_initial = I\_initial \* 0.2;

R\_initial = I\_initial \* 5;

V\_initial = N\_total \* 0.1;

S\_initial = N\_total - E\_initial - I\_initial - Q\_initial - J\_initial - T\_initial - V\_initial - R\_initial;

initial\_conditions = [S\_initial, E\_initial, I\_initial, Q\_initial, J\_initial, T\_initial, V\_initial, R\_initial];

% Simulate for the data period

tspan = [0 max(days)];

[t, y] = ode45(@(t, y) COVID19Model(t, y, params), tspan, initial\_conditions);

% Plot results

figure('Position', [100, 100, 1200, 600]);

% Plot actual vs. fitted cases

subplot(1, 2, 1);

scatter(days, cases, 100, 'b', 'filled');

hold on;

plot(t, y(:, 3), 'r-', 'LineWidth', 4);

title('COVID-19 Model Fit: Actual vs. Predicted Cases', 'FontSize', 23);

xlabel('Days (since Jul 9, 2021)', 'FontSize', 23);

ylabel('Number of Cases', 'FontSize', 23);

legend('Actual Data', 'Model Fit', 'FontSize', 16, 'Location', 'best');

grid on;

ax = gca;

ax.FontSize = 16;

ax.LineWidth = 1.5;

% Calculate and display fit metrics

I\_model = interp1(t, y(:, 3), days);

mse = mean((I\_model - cases).^2);

rmse = sqrt(mse);

mae = mean(abs(I\_model - cases));

r2 = 1 - sum((cases - I\_model).^2) / sum((cases - mean(cases)).^2);

subplot(1, 2, 2);

text(0.1, 0.8, sprintf('Goodness of Fit Metrics:'), 'FontSize', 18);

text(0.1, 0.7, sprintf('MSE: %.2f', mse), 'FontSize', 16);

text(0.1, 0.6, sprintf('RMSE: %.2f', rmse), 'FontSize', 16);

text(0.1, 0.5, sprintf('MAE: %.2f', mae), 'FontSize', 16);

text(0.1, 0.4, sprintf('R²: %.4f', r2), 'FontSize', 16);

axis off;

% Save optimized parameters for the forecast script

save('COVID19\_fit\_params.mat', 'params', 'initial\_conditions');

disp('Model fitting complete. Optimized parameters saved to COVID19\_fit\_params.mat');

%% COVID19Model Function

function dydt = COVID19Model(t, y, params)

% COVID19Model ODE function for simulating a compartmental COVID-19 model

%

% Inputs:

% t - Current time point

% y - Current state vector [S, E, I, Q, J, T, V, R]

% params - Structure containing model parameters

%

% Output:

% dydt - Derivatives of state variables

% Extract state variables

S = y(1); % Susceptible

E = y(2); % Exposed

I = y(3); % Infected

Q = y(4); % Quarantined

J = y(5); % Hospitalized

T = y(6); % Treated

V = y(7); % Vaccinated

R = y(8); % Recovered

% Total population

N = sum(y);

% Extract parameters from structure

Lambda = params.Lambda; % Recruitment rate

beta\_1 = params.beta\_1; % Infection rate from infected

beta\_2 = params.beta\_2; % Infection rate from hospitalized

chi\_1 = params.chi\_1; % Vaccination rate

chi\_2 = params.chi\_2; % Waning immunity rate from vaccination

psi\_1 = params.psi\_1; % Rate at which quarantined move to hospitalization

psi\_2 = params.psi\_2; % Rate at which quarantined return to susceptible

eta = params.eta; % Rate of loss of immunity

mu = params.mu; % Natural death rate

mu\_prime = params.mu\_prime; % Death rate for treated class

lambda = params.lambda; % Rate at which exposed become infected

phi = params.phi; % Rate at which exposed are quarantined

sigma\_1 = params.sigma\_1; % Rate at which infected are hospitalized

sigma\_2 = params.sigma\_2; % Rate at which infected are treated

delta\_1 = params.delta\_1; % Disease-induced death rate for infected

delta\_2 = params.delta\_2; % Disease-induced death rate for treated

delta\_3 = params.delta\_3; % Disease-induced death rate for hospitalized

gamma = params.gamma; % Rate at which hospitalized move to treatment

omega = params.omega; % Recovery rate of treated individuals

% Calculate a\_1 (force of infection)

a\_1 = (beta\_1 \* I + beta\_2 \* J) / N;

% System of differential equations

dydt = zeros(8, 1);

% dS/dt: Change in susceptible population

dydt(1) = Lambda - a\_1 \* S - chi\_1 \* S + chi\_2 \* V + psi\_2 \* Q + eta \* R - mu \* S;

% dE/dt: Change in exposed population

dydt(2) = a\_1 \* S - (lambda + phi + mu) \* E;

% dI/dt: Change in infected population

dydt(3) = lambda \* E - (sigma\_1 + sigma\_2 + delta\_1 + mu) \* I;

% dQ/dt: Change in quarantined population

dydt(4) = phi \* E - (psi\_1 + psi\_2 + mu) \* Q;

% dJ/dt: Change in hospitalized population

dydt(5) = psi\_1 \* Q + sigma\_1 \* I - (gamma + delta\_3 + mu) \* J;

% dT/dt: Change in treated population

dydt(6) = sigma\_2 \* I + gamma \* J - (omega + delta\_2 + mu\_prime) \* T;

% dV/dt: Change in vaccinated population

dydt(7) = chi\_1 \* S - (chi\_2 + mu) \* V;

% dR/dt: Change in recovered population

dydt(8) = omega \* T - (eta + mu) \* R;

end