```
function main()

    % Data Import
    Data = readtable('Dataset.xlsx',
'ReadVariableNames', true);
    Data = table2array(Data);

    % Extract variables
    t = Data(:, 1);      % time vector
    Y_true = Data(:, 2); % target variable
    X_all = Data(:, 3:7); % all influencing
parameters

    % Define lower and upper bounds for
```

```
parameters
    lb = [-100, -Inf, -pi, -Inf, -Inf, -pi, -Inf, -Inf,
-Inf, -Inf, 0, 0, 0, -inf, 0, -inf, 0, 0, 0];
    ub = [100, Inf, pi, Inf, Inf, pi, Inf, Inf, Inf,
Inf, 2, 5, 5, inf, 5, inf, 10, 10, 10];

    % Options for lsqnonlin
    options = optimoptions('lsqnonlin',
'Display', 'iter', 'MaxIterations', 500,
'FunctionTolerance', 1e-8,
'MaxFunctionEvaluations', 5000);

    % Initial guess for parameters
    initialGuess = [1, 1, pi/2, 0, 0.1, pi/2, 0,
0.5, 1, 1, 1, 1, 1, 1, 1, 0.7, 1, 0.1, 0.1];

    % Run optimization
    try
        % lsqnonlin optimization with updated
options
        fittedParams = lsqnonlin(@(b)
customModel(b, t, X_all, Y_true),
initialGuess, lb, ub, options);
```

```
        % Generate the fitted curve
        fittedCurve =
customModel(fittedParams, t, X_all,
Y_true) + Y_true;

        % Ensure fitted values do not exceed
26.5
        fittedCurve(fittedCurve > 16.0) = 16.0;

        % Plot results
        plotResults(t, Y_true, fittedCurve);

        % Display fitted parameters and
evaluation metrics
        displayResults(fittedParams, Y_true,
fittedCurve);

    catch
        disp('Error in lsqnonlin. Check initial
parameter values or model structure.');
    end
```

```
end

function result = customModel(params, t,
X_all, Y_true)
    result = params(1) +
params(2)*abs(sin(params(3)*t +
params(4))).^params(18) + ...
           params(5)*abs(cos(params(6)*t +
params(7))).^params(19) + ...
           params(8)*X_all(:, 1).^params(11) +
...
           params(9)*abs(X_all(:,
2)).^params(12) + ...
           params(10)*abs(X_all(:,
3)).^params(13) + ...
           params(11)*abs(X_all(:,
4)).^params(14) + ...
           params(12)*abs(X_all(:,
5)).^params(15);
end

function plotResults(t, Y_true, fittedCurve)
```

```
    % Plot the data and the fitted curve
    figure;
    plot(t, Y_true, 'b', 'DisplayName', 'True
Data');
    hold on;
    plot(t, fittedCurve, 'r', 'DisplayName',
'Fitted Curve');
    xlabel('Time');
    ylabel('Target Variable');
    legend('Location', 'best');
    title('Fitted Model');
    grid on;


end

function displayResults(fittedParams,
Y_true, fittedCurve)
    % Display fitted parameters and
evaluation metrics
    disp('Fitted Parameters:');
    disp(fittedParams);
```

```
    R_square = 1 - sum((Y_true -
fittedCurve).^2) / sum((Y_true -
mean(Y_true)).^2);
    MSE = mean((Y_true - fittedCurve).^2);
    RMSE = sqrt(MSE);
    PAME = (abs(Y_true - fittedCurve) /
mean(Y_true)) * 100;

    disp(['R-squared: ' num2str(R_square)]);
    disp(['Mean Squared Error (MSE): '
num2str(MSE)]);
    disp(['Root Mean Squared Error (RMSE):
' num2str(RMSE)]);
    disp(['Percentage of Absolute Mean
Error (PAME): ' num2str(mean(PAME)) '%']);
end
```