```
% Step 1: Generate sample data (latitude, longitude, and temperature)
num_samples = 200;
latitudes = -90 + 180 * rand(num_samples, 1); % Latitude range from -90 to 90
longitudes = -180 + 360 * rand(num_samples, 1); % Longitude range from -180 to 180

% Simulate some temperature data with a simple model
% Assume temperature is higher near the equator and varies with longitude
temperatures = 30 - 0.02 * (latitudes.^2) + 10 * cosd(longitudes) + 3 *
randn(num_samples, 1);

% Step 2: Define the kernel function
% Gaussian kernel function
kernel_function = @(x, xi, bandwidth) exp(-sum((x - xi).^2, 2) / (2 * bandwidth^2));

% Step 3: Implement the kernel regression function

% Step 4: Predict values on a grid to visualize the regression surface
grid_size = 50;
lat_range = linspace(min(latitudes), max(latitudes), grid_size);
lon_range = linspace(min(longitudes), max(longitudes), grid_size);
[Lat_grid, Lon_grid] = meshgrid(lat_range, lon_range);
Temp_grid = zeros(grid_size, grid_size);

bandwidth = 50; % Set the bandwidth for the kernel

n= num_samples;
coords=[latitudes, longitudes];
kernel_function = @(x, xi, bandwidth) exp(-sum((x - xi).^2, 2) / (2 * bandwidth^2));
K=eye(n)*0.1;
for i=1:n
    for j=1:n
        K(i,j)= K(i,j)+kernel_function(coords(i,:),coords(j,:),bandwidth);
    end
end
coeff= K\temperatures;

for i = 1:grid_size
    for j = 1:grid_size
        for k=1:n
            Temp_grid(i, j) = Temp_grid(i,j)+ coeff(k)*kernel_function(coords(k,:),
[Lat_grid(i, j), Lon_grid(i, j)],bandwidth);
        end
    end
end

% Step 5: Plot the original data and the regression surface
figure;
scatter3(latitudes, longitudes, temperatures, 'filled');
hold on;
mesh(Lat_grid, Lon_grid, Temp_grid, 'FaceAlpha', 0.5);
```

```
colorbar;
title('Kernel Regression of Temperature over Geographic Coordinates');
xlabel('Latitude');
ylabel('Longitude');
zlabel('Temperature (□C)');
hold off;


% Define a range of possible bandwidths to test
bandwidths = 5:5:150; % 1 to 30 with step size of 1

% Perform 5-fold cross-validation to select the best bandwidth
num_folds = 5;
[best_bandwidth, cv_mse] = select_bandwidth_via_cv([latitudes, longitudes],
temperatures, num_folds, bandwidths);

% Now use the best bandwidth to predict the entire grid
Temp_grid_cv = zeros(grid_size, grid_size);


%for i = 1:grid_size
%    for j = 1:grid_size
%        Temp_grid_cv(i, j) = kernel_regression(coords, coeff, [Lat_grid(i, j),
Lon_grid(i, j)], best_bandwidth);
%    end
%end

% Plot the original data and the regression surface with the best bandwidth
figure;
scatter3(latitudes, longitudes, temperatures, 'filled');
hold on;
mesh(Lat_grid, Lon_grid, Temp_grid_cv, 'FaceAlpha', 0.5);
colorbar;
title(sprintf('Kernel Regression with Cross-Validation (Bandwidth = %d)',
best_bandwidth));
xlabel('Latitude');
ylabel('Longitude');
zlabel('Temperature (□C)');
hold off;

% Plot the cross-validation results
figure;
plot(bandwidths, cv_mse, '-o');
xlabel('Bandwidth');
ylabel('Cross-validated MSE');
title('Cross-validation Results for Bandwidth Selection');
grid on;


function temperature_pred = kernel_regression(coords, temperatures, coord_new,
bandwidth)
```

```
    kernel_function = @(x, xi, bandwidth) exp(-sum((x - xi).^2, 2) / (2 *
bandwidth^2));



    n= length(temperatures);
    K=eye(n)*.1;
    for i=1:n
        for j=1:n
            K(i,j)= K(i,j)+kernel_function(coords(i,:),coords(j,:),bandwidth);
        end
    end
    coeff= K\temperatures;

    temperature_pred=0;
    for k=1:n
         temperature_pred= temperature_pred+
coeff(k)*kernel_function(coords(k,:),coord_new ,bandwidth);
    end



end




function [best_bandwidth, cv_mse] = select_bandwidth_via_cv(coords, temps, num_folds,
bandwidths)
    cv_mse = zeros(length(bandwidths), 1); % To hold the mean squared error for each
bandwidth

    % Split data into k folds
    indices = crossvalind('Kfold', length(temps), num_folds);

    % Iterate over all bandwidths
    for b = 1:length(bandwidths)


        mse_folds = zeros(num_folds, 1);

        % Perform k-fold cross-validation
        for fold = 1:num_folds
            % Create training and validation sets
            test_idx = (indices == fold);
            train_idx = ~test_idx;

            coords_train = coords(train_idx, :);
```

```
            temps_train = temps(train_idx);
            coords_test = coords(test_idx, :);
            temps_test = temps(test_idx);

            % Predict temperatures using kernel regression with current bandwidth
            temps_pred = arrayfun(@(i) kernel_regression(coords_train, temps_train,
coords_test(i,:), bandwidths(b)), 1:sum(test_idx));

            % Calculate and store the mean squared error for this fold
            mse_folds(fold) = mean((temps_test - temps_pred').^2);
        end

        % Calculate the mean of the mean squared error across all folds for this
bandwidth
        cv_mse(b) = mean(mse_folds);
    end

    % Select the bandwidth with the smallest cross-validated MSE
    [~, min_idx] = min(cv_mse);
    best_bandwidth = bandwidths(min_idx);
end
```