

Contents

- [Step 1: Import CSV Data](#)
- [Step 2: R Bridge Implementation](#)
- [Step 3a: MDFT Formulation to Calculate Preference Dynamics in Parallel](#)
- [Step 3b: MDFT Formulation with State Continuity](#)
- [Step 4: Output Results](#)
- [Helper Functions](#)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main Simulation Script for BoundingOverwatch Project with R Integration
% Randomly 10 trials is used to validate predictions
% DFT Parameters:
% phi1 - sensitivity to attribute differences (typically 0.5-2)
% phi2 - memory/feedback strength (0-1)
% tau - decision time steps (integer > 0)
% error_sd - noise standard deviation (sigma_e)
% beta - attribute weights from R estimation
% w - attention weights (default [0.5;0.5])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;
clear all;
```

Step 1: Import CSV Data

(reference apolloMain_5 amd apolloMain_6 as example for data manipulation) biasData = readtable('user_choices.csv'); % Replace with the path to your data file disp('User bias data imported successfully.');

taskChoice_Data = readtable('user_choices.csv'); % Replace with the path to your data file disp('User task choice data imported successfully.');

```
robotChoice_Data = readtable('G:\My Drive\myResearch\Research Experimentation\Apollo\apollo\data\Bounding_Overwatch_Data\HumanData_Bounding_Overwatch - 20Split.csv')
% Convert all column headers to lowercase
robotChoice_Data.Properties.VariableNames = lower(robotChoice_Data.Properties.VariableNames);
disp('User robot choice data imported successfully.');
```

% Randomly select 10 rows (or all rows if fewer than 10)

```
numRows = height(robotChoice_Data);
randomIndices = randperm(numRows, min(10, numRows));
robotChoice_Data = robotChoice_Data(randomIndices, :);
```

% Extract robot state attributes dynamically

```
robot_states = struct();
attributeSuffixes = {'traversability', 'visibility'}; % No leading underscores
for i = 1:3
    attr = attributeSuffixes
    csvColName = sprintf('robot%d_%s', i, attr{1}); % Matches CSV column names
    structFieldName = attr{1}; % Valid field name
    if ismember(csvColName, robotChoice_Data.Properties.VariableNames)
        robot_states.([ 'robot' num2str(i) ]). (structFieldName) = robotChoice_Data.(csvColName);
    else
        warning('Missing attribute column: %s', csvColName);
        robot_states.([ 'robot' num2str(i) ]). (structFieldName) = NaN(height(robotChoice_Data), 1);
    end
end
end
```

% Extract choice data and other metadata

```
choices = robotChoice_Data.choice;
participant_ids = robotChoice_Data.id;
stake_types = robotChoice_Data.stakes;
time_spent = robotChoice_Data.timeelapsed;
```

User robot choice data imported successfully.

Step 2: R Bridge Implementation

```
disp('Initializing R bridge...');
```

% Configure paths

```
rscript_path = 'C:\Program Files\R\R-4.4.2\bin\x64\Rscript.exe';
r_script = 'G:\My Drive\myResearch\Research Experimentation\Apollo\apollo\example\DFT_Bounding_Overwatch.R';
csvFile = 'G:\My Drive\myResearch\Research Experimentation\Apollo\apollo\data\Bounding_Overwatch_Data\HumanData_Bounding_Overwatch - 80Split.csv';
outputDir = 'G:\My Drive\myResearch\Research Experimentation\Apollo\apollo\Output_BoundingOverwatch';
```

% Verify installations

```
if ~isfile(rscript_path)
    error('Rscript.exe not found at: %s', rscript_path);
elseif ~isfile(r_script)
    error('R script not found at: %s', r_script);
elseif ~isfile(csvFile)
    error('Input CSV not found at: %s', csvFile);
```

```

elseif ~isfolder(outputDir)
    warning('Output folder does not exist, creating: %s', outputDir);
    mkdir(outputDir);
end

% Execute R with JSON output
try
    % Use proper argument formatting
    cmd = sprintf(['"%s" "%s" ', ...
        '-i "%s" -o "%s"', ...
        rscript_path, r_script, csvFile, outputDir]);

    [status,result] = system(cmd);

    if status == 0
        % Handle output path (whether directory or file)
        if isfolder(outputDir)
            jsonFile = fullfile(outputDir, 'DFT_output.json');
        else
            jsonFile = outputDir;
        end

        % Parse JSON output
        if exist(jsonFile, 'file')
            jsonText = fileread(jsonFile);
            params = jsondecode(jsonText);

            % Extract parameters with validation
            %Boundedphi1, phi2 parameters
            phi1 = min(max(0, validateParam(params, 'phi1', 0.5)),5); % Ensure non-negative
            phi2 = min(max(0, validateParam(params, 'phi2', 0.8)), 0.99); % Constrain 0-1
            %tau = min(1 + exp(validateParam(params, 'timesteps', 0.5)),100); %Constrain to 100

            %Raw phi1, phi2 parameters
            %phi1 = validateParam(params, 'phi1', 0.5);
            %phi2 = validateParam(params, 'phi2', 0.8);
            tau = 1 + exp(validateParam(params, 'timesteps', 0.5));
            error_sd = min(max(0.1, validateParam(params, 'error_sd', 0.1)), 1); % still clip here

            % Extract attribute weights
            beta_weights = [
                params.b_attr1;
                params.b_attr2;
                params.b_attr3;
                params.b_attr4
            ];

            % Get initial preferences from ASCs
            initial_P = [
                validateParam(params, 'asc_1', 0);
                validateParam(params, 'asc_2', 0);
                validateParam(params, 'asc_3', 0);
            ];

            disp('Estimated Parameters:');
            disp(['phi1: ', num2str(phi1)]);
            disp(['phi2: ', num2str(phi2)]);
            disp(['tau: ', num2str(tau)]);
            disp(['error_sd: ', num2str(error_sd)]);
            disp('Initial Preferences (from ASCs):');
            disp(initial_P);
        else
            error('R output file not found');
        end
    else
        error('R execution failed: %s', result);
    end
catch ME
    disp('Error during R execution:');
    disp(getReport(ME, 'extended'));
    [phi1, phi2, tau, error_sd] = getFallbackParams();
    beta_weights = [0.3; 0.2; 0.4; 0.5]; % Default weights
    initial_P = zeros(3,1); % Neutral initial preferences
end

```

Initializing R bridge...

Step 3a: MDFT Formulation to Calculate Preference Dynamics in Parallel

```

%%{
% (MDFT calculations based on estimated parameters)
% Create M matrix from current trial's attributes
% C11-C14 are consequence attributes for Robot 1
% C21-C24 are consequence attributes for Robot 2

```

```

% C31-C34 are consequence attributes for Robot 3
for current_trial = 1:height(robotChoice_Data)
    num_attributes = 4;

    M = [
        robotChoice_Data.c11(current_trial), robotChoice_Data.c12(current_trial), robotChoice_Data.c13(current_trial), robotChoice_Data.c14(current_trial);
        robotChoice_Data.c21(current_trial), robotChoice_Data.c22(current_trial), robotChoice_Data.c23(current_trial), robotChoice_Data.c24(current_trial);
        robotChoice_Data.c31(current_trial), robotChoice_Data.c32(current_trial), robotChoice_Data.c33(current_trial), robotChoice_Data.c34(current_trial)
    ];

    % Normalize M values by dividing by 2 and clamping to [0.01, 1]
    %{
    M = M / 2;
    M = max(0.01, min(1, M));
    %}

    % --- Global Max Normalization ---
    %{
    global_max = max(robotChoice_Data(:, {'c11','c12','c13','c14','c21','c22','c23','c24','c31','c32','c33','c34'}), [], 'all', 'omitnan');
    if ~isfinite(global_max) || global_max <= 0
        global_max = 1; % fallback in case of zero or NaN
    end

    M = M / global_max; % Normalize by global max
    M = max(0.01, min(1, M)); % Clamp to [0.01, 1]
    %}

    % --- Row-wise Min-Max Normalization ---
    %%{
    for i = 1:size(M, 1)
        row = M(i, :);
        min_val = min(row);
        max_val = max(row);

        if max_val == min_val
            M(i, :) = pmax(0.01, pmin(1, row)); % constant row: clamp only
        else
            norm_row = (row - min_val) / (max_val - min_val);
            M(i, :) = max(0.01, min(1, norm_row)); % clamp to [0.01, 1]
        end
    end
    %%}

    attributes = {'C1 - Easy Nav, Low Exposure', 'C2 - Hard Nav, Low Exposure', 'C3 - Easy Nav, High Exposure', 'C4 - Hard Nav, High Exposure'};
    beta = beta_weights ./ sum(abs(beta_weights));
    beta = beta';

    [E_P, V_P, choice_probs, P_tau] = calculateDFTdynamics(...
        phi1, phi2, tau, error_sd, beta, M, initial_P);

    % Display results for the frame
    disp('=== Trial Analysis ===');
    disp(['Trial: ', num2str(current_trial)]);
    disp(['Participant: ', num2str(participant_ids(current_trial))]);
    disp(['Actual Choice: Robot ', num2str(choices(current_trial))]);

    disp('M matrix (alternatives x attributes):');
    disp(array2table(M, ...
        'RowNames', {'Robot1','Robot2','Robot3'}, ...
        'VariableNames', attributes));

    disp('DFT Results:');
    disp(['E_P: ', num2str(E_P', '%.2f ')]);
    disp(['Choice probabilities: ', num2str(choice_probs', '%.3f ')]);
    [~, predicted_choice] = max(choice_probs);
    disp(['Predicted choice: Robot ', num2str(predicted_choice)]);
    disp(['Actual choice: Robot ', num2str(choices(current_trial))]);
    disp(' ');

    if predicted_choice == choices(current_trial)
        disp('✓ Prediction matches actual choice');
    else
        disp('X Prediction differs from actual choice');
    end

    % Plot evolution
    figure;
    %plot(0:tau, P_tau);
    % Replace the plotting section with:
    tau_rounded = round(tau); % Ensure integer steps
    if size(P_tau,2) == tau_rounded+1 % Validate dimensions
        plot(0:tau_rounded, P_tau);
    else
        warning('Dimension mismatch: P_tau has %d cols, expected %d',...
            size(P_tau,2), tau_rounded+1);
    end
end

```

```

        plot(P_tau'); % Fallback plot
    end
    xlabel('Preference Step (\tau)');
    ylabel('Preference Strength');
    legend({'Robot1','Robot2','Robot3'});
    title(sprintf('Preference Evolution (Trial %d)', current_trial));
    grid on;
end
%%}

```

=== Trial Analysis ===

Trial: 1

Participant: 141831

Actual Choice: Robot 3

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1	0.57827	0.42173	0.01
Robot2	1	0.6312	0.3688	0.01
Robot3	1	0.54961	0.45039	0.01

DFT Results:

E_P: 0.00 0.02 -0.02

Choice probabilities: 0.339 0.392 0.269

Predicted choice: Robot 2

Actual choice: Robot 3

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 2

Participant: 141831

Actual Choice: Robot 2

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1	0.44957	0.55043	0.01
Robot2	1	0.5533	0.4467	0.01
Robot3	1	0.50618	0.49382	0.01

DFT Results:

E_P: -0.01 0.02 -0.01

Choice probabilities: 0.301 0.399 0.300

Predicted choice: Robot 2

Actual choice: Robot 2

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 3

Participant: 125802

Actual Choice: Robot 3

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1	0.49468	0.50532	0.01
Robot2	1	0.46147	0.53853	0.01
Robot3	1	0.52825	0.47175	0.01

DFT Results:

E_P: 0.01 -0.00 -0.00

Choice probabilities: 0.350 0.321 0.329

Predicted choice: Robot 1

Actual choice: Robot 3

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 4

Participant: 141831

Actual Choice: Robot 3

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1	0.50762	0.49238	0.01
Robot2	1	0.54112	0.45888	0.01
Robot3	1	0.52591	0.47409	0.01

DFT Results:

E_P: 0.00 0.01 -0.01

Choice probabilities: 0.333 0.366 0.301

Predicted choice: Robot 2

Actual choice: Robot 3

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 5

Participant: 125802

Actual Choice: Robot 3

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1	0.55138	0.44862	0.01
Robot2	1	0.48487	0.51513	0.01
Robot3	1	0.50079	0.49921	0.01

DFT Results:

E_P: 0.02 -0.00 -0.01

Choice probabilities: 0.386 0.325 0.289

Predicted choice: Robot 1

Actual choice: Robot 3

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 6

Participant: 141831

Actual Choice: Robot 3

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1	0.50662	0.49338	0.01
Robot2	1	0.50436	0.49564	0.01
Robot3	1	0.53586	0.46414	0.01

DFT Results:

E_P: 0.00 0.00 -0.00

Choice probabilities: 0.342 0.341 0.317

Predicted choice: Robot 1

Actual choice: Robot 3

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 7

Participant: 141831

Actual Choice: Robot 1

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1	0.51739	0.48261	0.01
Robot2	1	0.48096	0.51904	0.01
Robot3	1	0.47196	0.52804	0.01

DFT Results:

E_P: 0.01 0.00 -0.01

Choice probabilities: 0.374 0.341 0.285

Predicted choice: Robot 1

Actual choice: Robot 1

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 8

Participant: 125802

Actual Choice: Robot 1

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1	0.52664	0.47336	0.01
Robot2	1	0.54328	0.45672	0.01
Robot3	1	0.50397	0.49603	0.01

DFT Results:

E_P: 0.01 0.01 -0.01

Choice probabilities: 0.350 0.368 0.283

Predicted choice: Robot 2

Actual choice: Robot 1

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 9

Participant: 125802

Actual Choice: Robot 3

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1	0.50479	0.49521	0.01
Robot2	1	0.55569	0.44431	0.01
Robot3	1	0.48468	0.51532	0.01

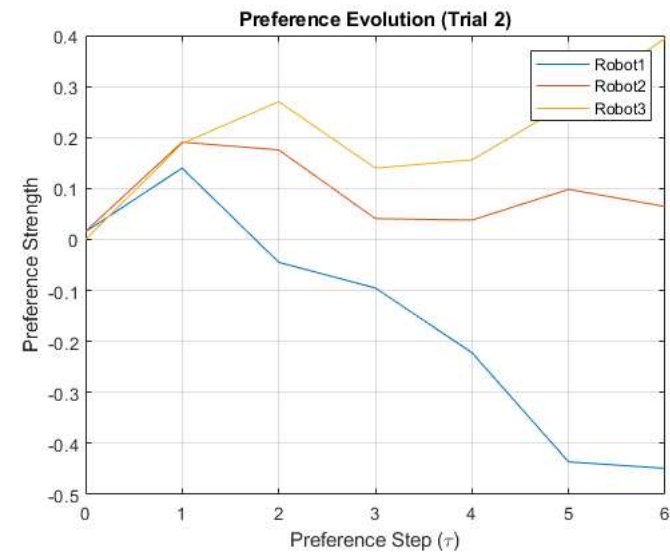
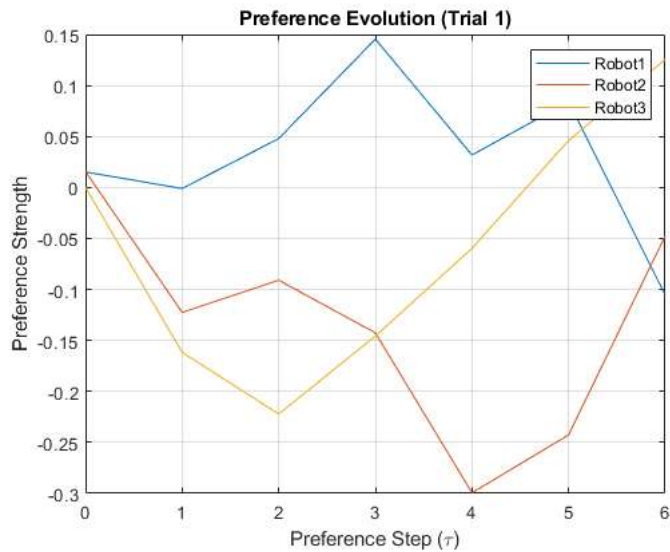
DFT Results:
E_P: 0.00 0.02 -0.02
Choice probabilities: 0.337 0.388 0.274
Predicted choice: Robot 2
Actual choice: Robot 3

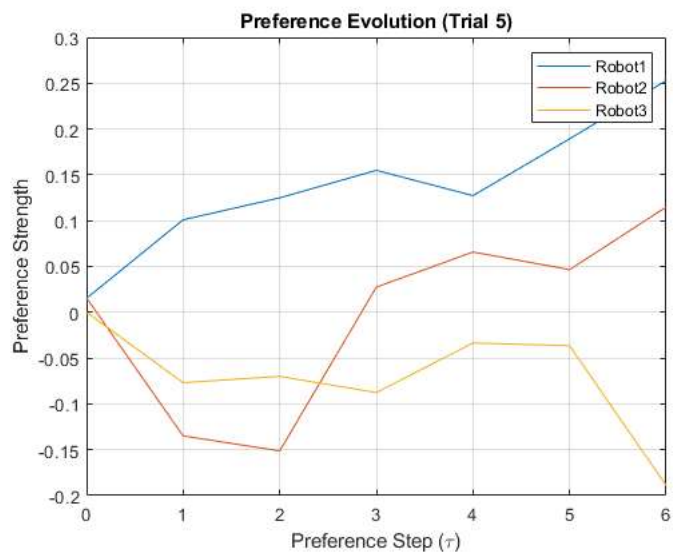
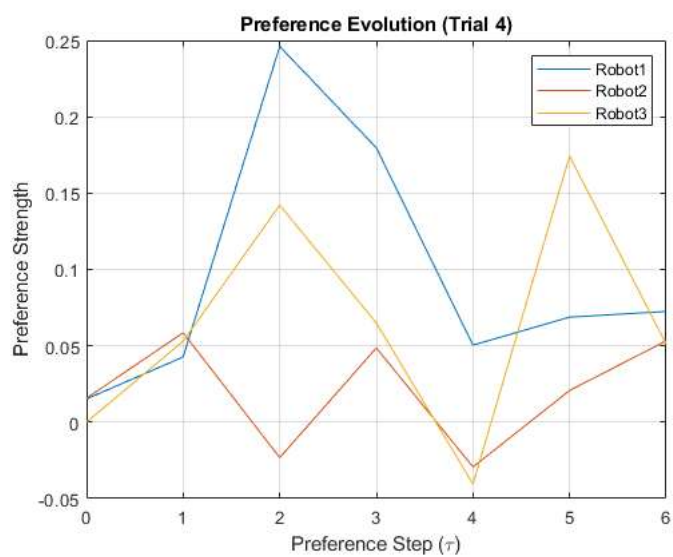
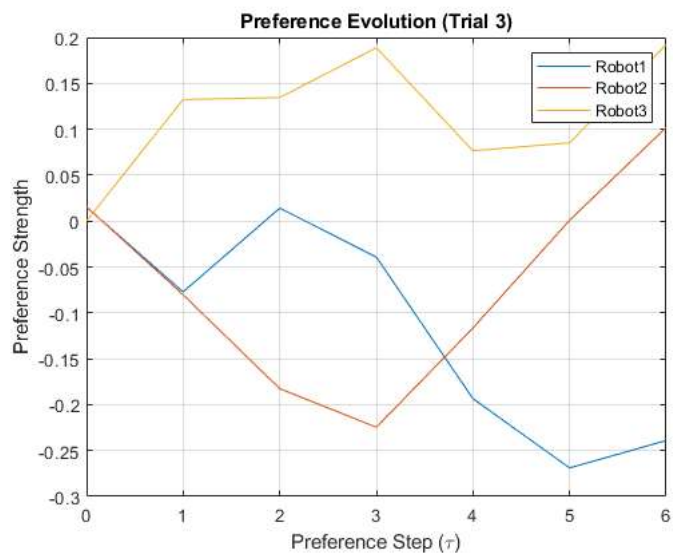
X Prediction differs from actual choice
=== Trial Analysis ===
Trial: 10
Participant: 125802
Actual Choice: Robot 1

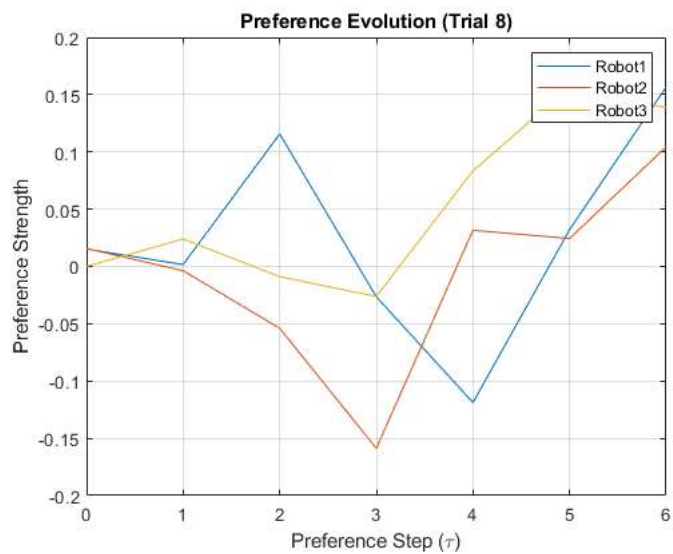
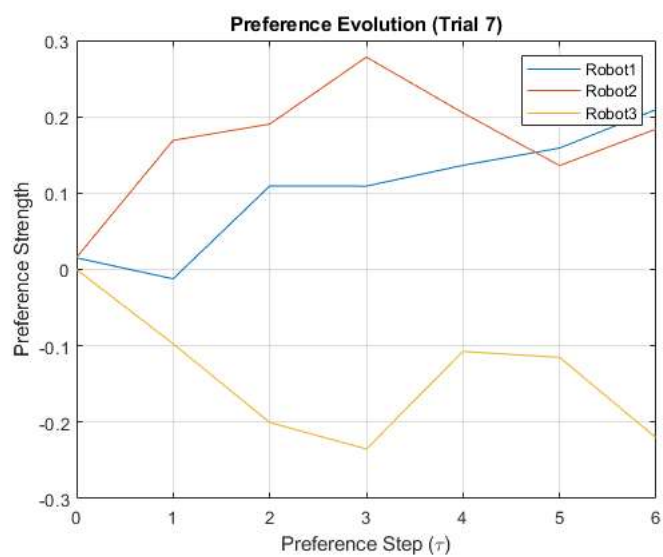
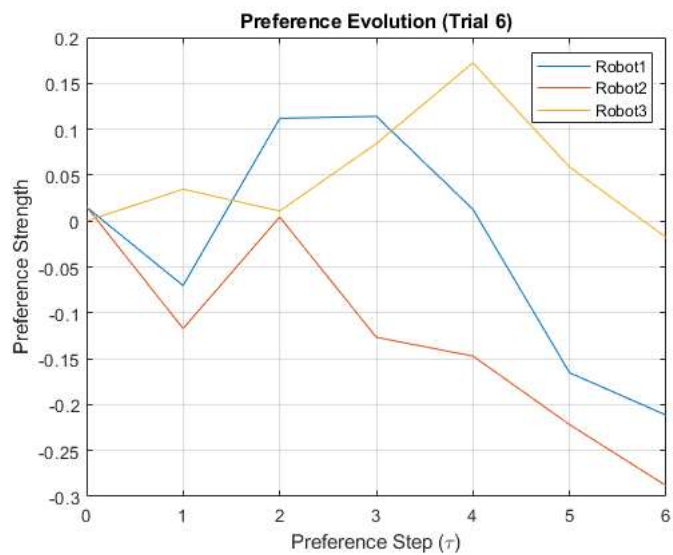
M matrix (alternatives x attributes):				
	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1	0.40645	0.59355	0.01
Robot2	1	0.48832	0.51168	0.01
Robot3	1	0.47183	0.52817	0.01

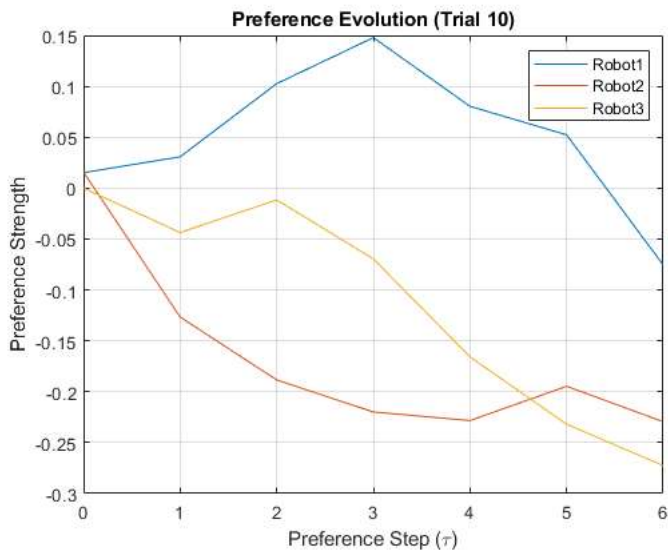
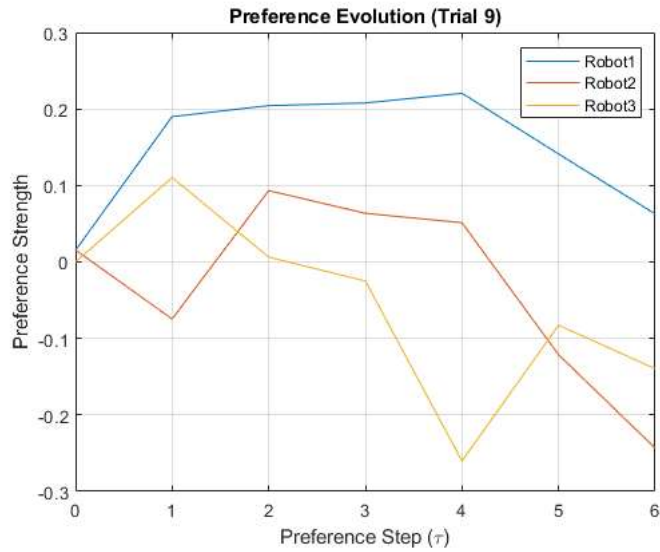
DFT Results:
E_P: -0.01 0.02 -0.00
Choice probabilities: 0.305 0.382 0.312
Predicted choice: Robot 2
Actual choice: Robot 1

X Prediction differs from actual choice









Step 3b: MDFT Formulation with State Continuity

```
%{
% Initialize preference state tracking
if ~exist('P_final_prev', 'var')
    P_final_prev = initial_P; % Use estimated initial preferences for first trial
end

for current_trial = 1:height(robotChoice_Data)
    % Create M matrix for current trial
    M = [
        robotChoice_Data.c11(current_trial), robotChoice_Data.c12(current_trial), robotChoice_Data.c13(current_trial), robotChoice_Data.c14(current_trial);
        robotChoice_Data.c21(current_trial), robotChoice_Data.c22(current_trial), robotChoice_Data.c23(current_trial), robotChoice_Data.c24(current_trial);
        robotChoice_Data.c31(current_trial), robotChoice_Data.c32(current_trial), robotChoice_Data.c33(current_trial), robotChoice_Data.c34(current_trial)
    ];

    % Normalize beta weights
    beta = beta_weights ./ sum(abs(beta_weights));

    % Calculate DFT dynamics using previous trial's final state
    [E_P, V_P, choice_probs, P_tau] = calculateDFTdynamics(...
        phi1, phi2, tau, error_sd, beta, M, P_final_prev);

    % Store final preference state for next trial
    P_final_prev = P_tau(:, end);

    % Display results
    disp('=== Trial Analysis ===');
    disp(['Trial: ', num2str(current_trial)]);
    disp(['Participant: ', num2str(participant_ids(current_trial))]);
    disp(['Actual Choice: Robot ', num2str(choices(current_trial))]);

    disp('Initial Preferences (from previous trial):');
    disp(array2table(P_tau(:,1), 'VariableNames', {'Robot1','Robot2','Robot3'})); % Fixed this line
end
}
```

```

disp('Final Preferences:');
disp(array2table(P_tau(:,end)', 'VariableNames', {'Robot1','Robot2','Robot3'})); % Fixed this line

% Enhanced plotting with initial/final state markers
figure;
plot(0:tau, P_tau, 'LineWidth', 2);
hold on;
% Mark initial state
scatter(zeros(3,1), P_tau(:,1), 100, 'filled');
% Mark final state
scatter(tau*ones(3,1), P_tau(:,end), 100, 'x', 'LineWidth', 2);
hold off;

xlabel('Preference Step (\tau)');
ylabel('Preference Strength');
legend({'Robot1','Robot2','Robot3','Initial State','Final State'});
title(sprintf('Preference Evolution (Trial %d)', current_trial));
grid on;
end
%}

```

Step 4: Output Results

```

disp('Saving results to CSV...');
output_table = table(E_P, V_P, P_tau(end,:), ...
    'VariableNames', {'ExpectedPreference', 'VariancePreference', 'FinalPreferences'});
writetable(output_table, 'results.csv');
disp('Results saved successfully!');
%}

```

Saving results to CSV...

Error using table
All table variables must have the same number of rows.

Error in main_BoundingOverwatch (line 302)
output_table = table(E_P, V_P, P_tau(end,:), ...

Helper Functions

```

function param = validateParam(params, name, default)
    if isfield(params, name) && isnumeric(params.(name))
        param = params.(name);
    else
        warning('Using default for %s', name);
        param = default;
    end
end

function [phi1, phi2, tau, error_sd] = getFallbackParams()
    phi1 = 0.5 + 0.1*randn();
    phi2 = 0.8 + 0.1*randn();
    tau = 10 + randi(5);
    error_sd = 0.1 + 0.05*rand();
    warning('Using randomized default parameters');
end

```

Estimated Parameters:
phi1: 3.499
phi2: 0.1
tau: 5.9998
error_sd: 0.1
Initial Preferences (from ASCs):
0.0152 0.0157 0