

## Contents

- [Step 1: Import CSV Data](#)
- [Step 2: R Bridge Implementation](#)
- [Step 3: MDTF Formulation to Calculate Preference Dynamics](#)
- [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_epsilon)
% 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 3: MDFT Formulation to Calculate Preference Dynamics

(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 trial
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);
    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
%{
%% 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!');
%}

```

=== Trial Analysis ===

Trial: 1

Participant: 125802

Actual Choice: Robot 2

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	1.8542	1.0334	1.0062	0.18542
Robot2	1.5482	0.8474	0.85561	0.15482
Robot3	1.6534	0.90354	0.91523	0.16534

DFT Results:

E\_P: -3.39 1.47 1.88

Choice probabilities: 0.000 0.016 0.984

Predicted choice: Robot 3

Actual choice: Robot 2

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 2

Participant: 125802

Actual Choice: Robot 2

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	1.6771	0.94661	0.89815	0.16771
Robot2	1.5779	0.90195	0.83376	0.15779
Robot3	1.75	1.0285	0.89644	0.175

DFT Results:

E\_P: -1.29 5.56 -4.26

Choice probabilities: 0.000 1.000 0.000

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 x attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1.1754	0.57957	0.71336	0.11754
Robot2	1.2804	0.63962	0.76887	0.12804
Robot3	1.3269	0.69174	0.76789	0.13269

DFT Results:

E\_P: 6.36 -2.20 -4.14

Choice probabilities: 1.000 0.000 0.000

Predicted choice: Robot 1

Actual choice: Robot 3

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 4

Participant: 125802

Actual Choice: Robot 3

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	1.4994	0.894	0.75533	0.14994
Robot2	1.5805	0.84776	0.8908	0.15805
Robot3	1.6969	0.93449	0.93208	0.16969

DFT Results:

E\_P: 0.52 6.46 -7.02

Choice probabilities: 0.000 1.000 0.000

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	1.8598	1.0972	0.94855	0.18598
Robot2	1.8077	1.0814	0.90712	0.18077
Robot3	1.7338	1.0234	0.88385	0.17338

DFT Results:

E\_P: -4.02 -4.46 8.49

Choice probabilities: 0.000 0.000 1.000

Predicted choice: Robot 3

Actual choice: Robot 3

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 6

Participant: 125802

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	1.3164	0.79078	0.65727	0.13164
Robot2	1.0476	0.56137	0.59098	0.10476
Robot3	1.0471	0.51085	0.64092	0.10471

DFT Results:

E\_P: -3.26 5.90 -2.70

Choice probabilities: 0.000 1.000 0.000

Predicted choice: Robot 2

Actual choice: Robot 2

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 7

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	1.6791	0.98568	0.86131	0.16791
Robot2	1.7772	1.1044	0.85055	0.17772
Robot3	1.6603	0.98705	0.83924	0.16603

DFT Results:

E\_P: -11.17 -5.20 16.35

Choice probabilities: 0.000 0.000 1.000

Predicted choice: Robot 3

Actual choice: Robot 3

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 8

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	1.5868	0.98673	0.75877	0.15868
Robot2	1.4619	0.8791	0.72897	0.14619
Robot3	1.459	0.89355	0.71138	0.1459

DFT Results:

E\_P: -6.72 -0.86 7.56

Choice probabilities: 0.000 0.000 1.000

Predicted choice: Robot 3

Actual choice: Robot 3

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 9

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	1.3187	0.76671	0.68383	0.13187
Robot2	0.98683	0.46154	0.62398	0.098683

Robot3 0.93147 0.46242 0.56219 0.093147

DFT Results:  
E\_P: -2.06 -1.82 3.85  
Choice probabilities: 0.000 0.000 1.000  
Predicted choice: Robot 3  
Actual choice: Robot 1

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 10

Participant: 141831

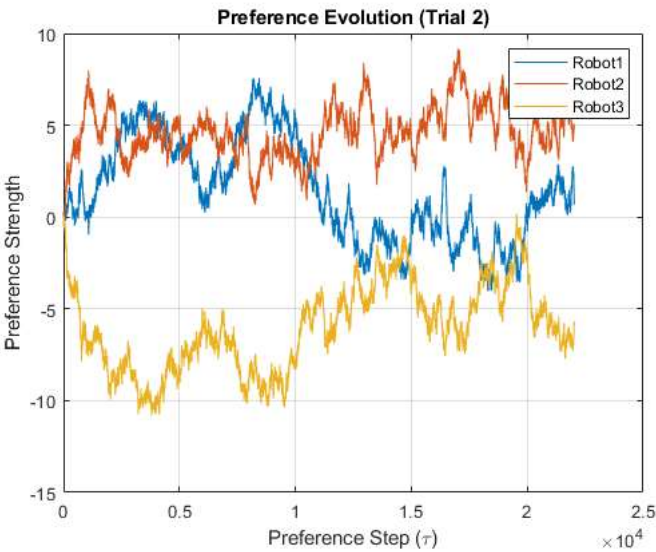
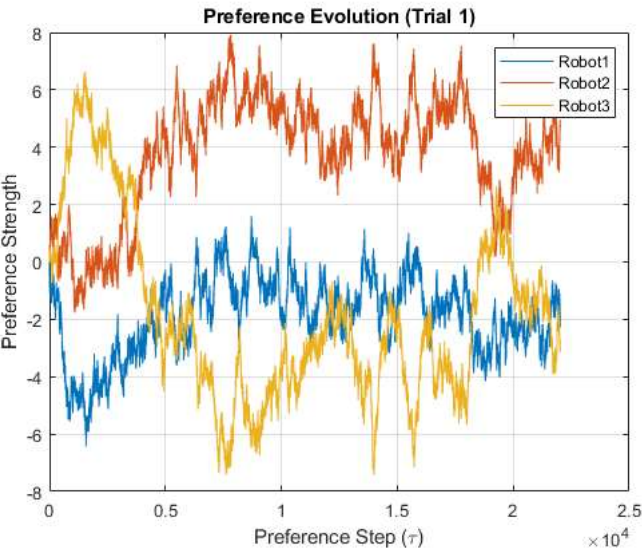
Actual Choice: Robot 3

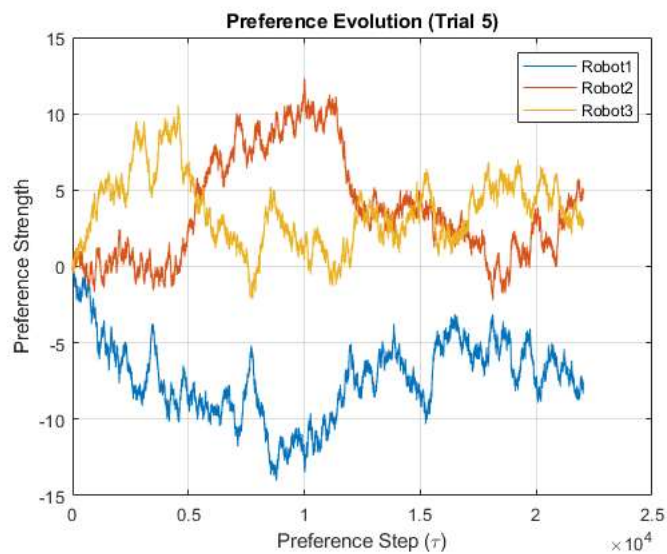
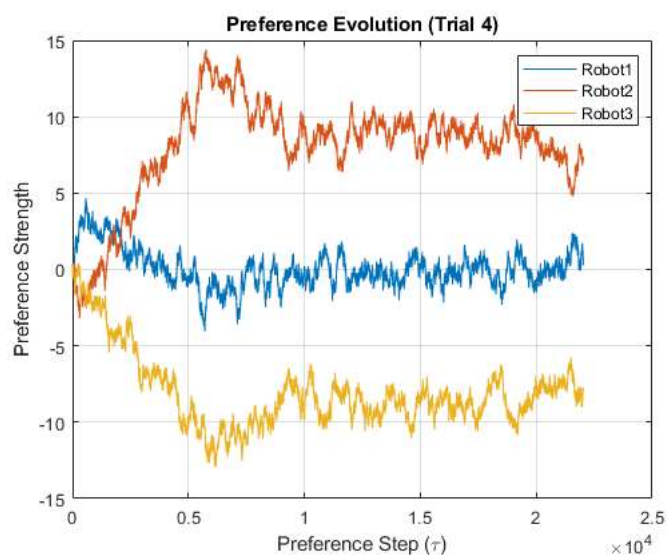
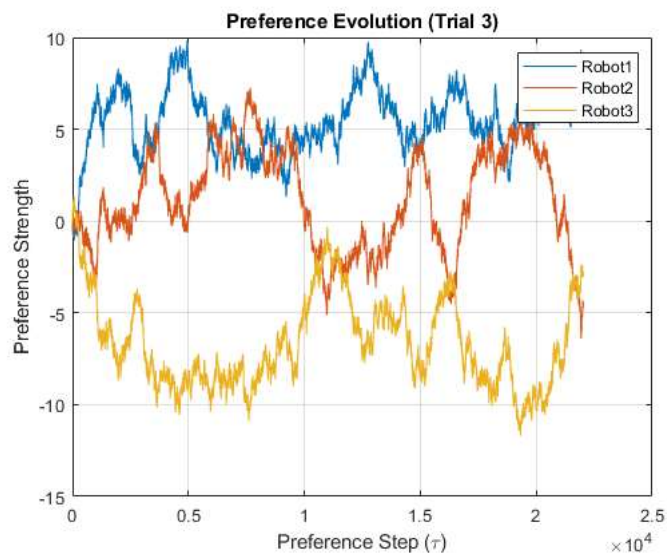
M matrix (alternatives x attributes):

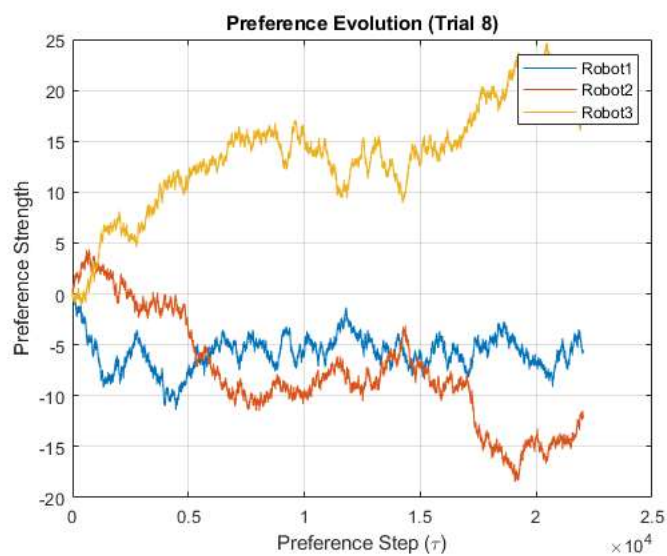
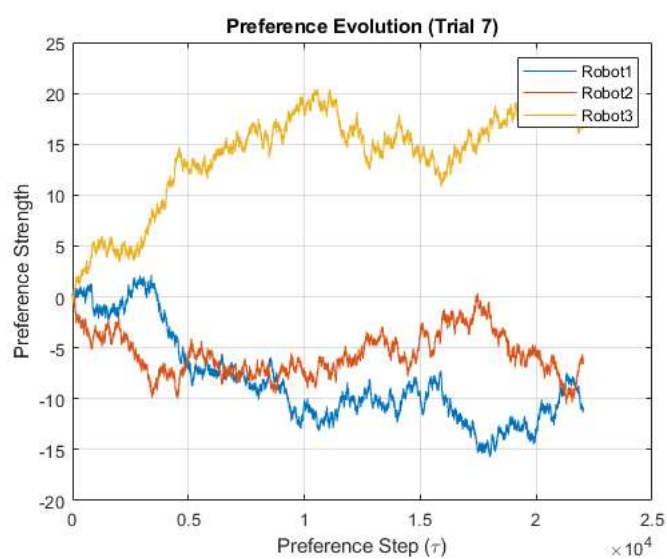
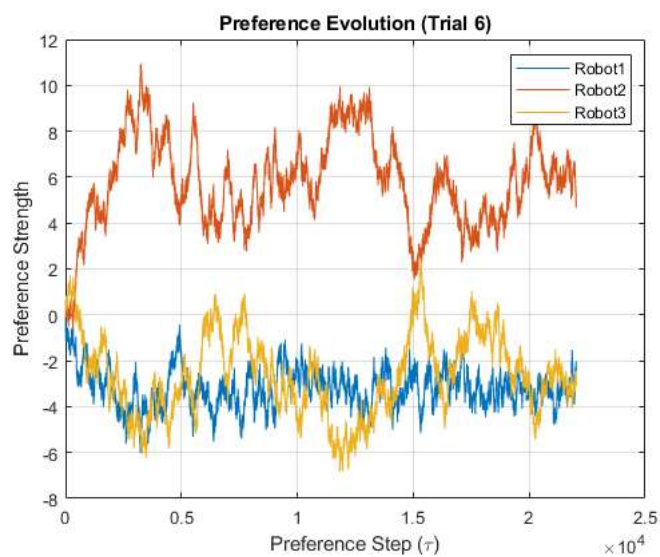
	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	1.7507	1.045	0.88081	0.17507
Robot2	1.674	0.96451	0.87689	0.1674
Robot3	1.7767	1.0672	0.88716	0.17767

DFT Results:  
E\_P: 3.83 5.62 -9.45  
Choice probabilities: 0.000 1.000 0.000  
Predicted choice: Robot 2  
Actual choice: Robot 3

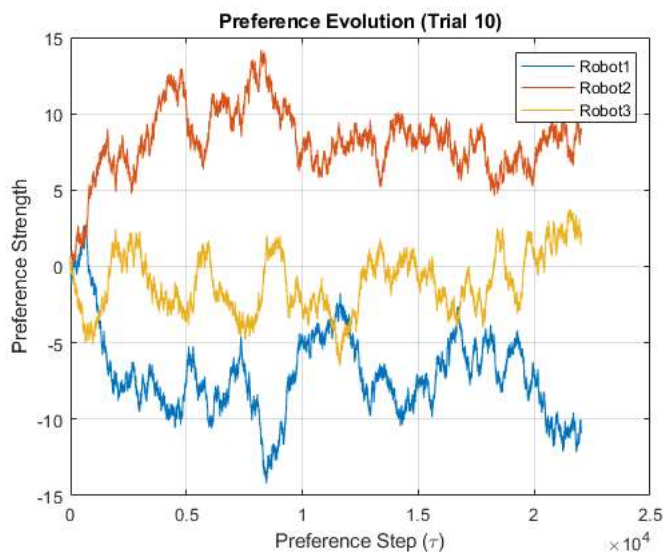
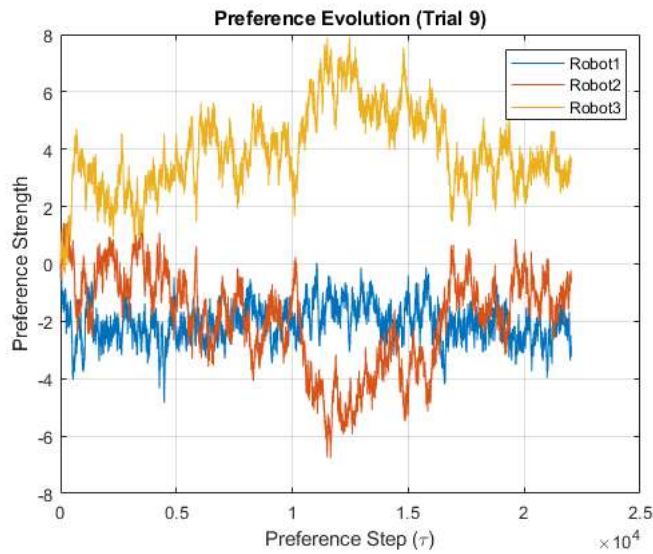
X Prediction differs from actual choice











## 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: 6.3422

phi2: 0.1

tau: 22027.4658

error\_sd: 0.1

Initial Preferences (from ASCs):

0.0236    0.0289    0

