

## Contents

- [Step 1: Import CSV Data](#)
- [Step 2: R Bridge Implementation](#)
- [Step 3: MDFT 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.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
    for 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.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 = max(0, validateParam(params, 'phi1', 0.5)); % Ensure non-negative
            phi2 = min(max(0, validateParam(params, 'phi2', 0.8)), 1); % Constrain 0-1

            %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));

    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);
    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: 181700
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	0.76061	0.75914	0.13816
Robot2	0.76209	0.41111	0.42719	0.076209
Robot3	1	0.65172	0.575	0.11152

```
DFT Results:
E_P: -149.54 236.54 -87.10
Choice probabilities: 0.000 1.000 0.000
Predicted choice: Robot 2
Actual choice: Robot 2
```

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 2

Participant: 214504

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	0.37014	0.21347	0.19368	0.037014
Robot2	0.3567	0.21128	0.18109	0.03567
Robot3	0.59097	0.3893	0.26076	0.059097

DFT Results:

E\_P: 85.50 105.97 -191.57

Choice probabilities: 0.000 1.000 0.000

Predicted choice: Robot 2

Actual choice: Robot 1

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 3

Participant: 124737

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	0.89528	0.54714	0.43768	0.089528
Robot2	0.93259	0.56376	0.46209	0.093259
Robot3	0.87915	0.5069	0.46017	0.087915

DFT Results:

E\_P: 14.68 -37.86 23.08

Choice probabilities: 0.000 0.000 1.000

Predicted choice: Robot 3

Actual choice: Robot 3

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 4

Participant: 142426

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	0.78613	0.43592	0.42882	0.078613
Robot2	0.78639	0.42333	0.44171	0.078639
Robot3	0.79292	0.45043	0.42179	0.079292

DFT Results:

E\_P: 3.47 -2.48 -1.10

Choice probabilities: 0.987 0.003 0.010

Predicted choice: Robot 1

Actual choice: Robot 1

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 5

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	0.6582	0.39539	0.32864	0.06582
Robot2	0.52379	0.28068	0.29549	0.052379
Robot3	0.52353	0.25542	0.32046	0.052353

DFT Results:

E\_P: -106.72 58.52 48.09

Choice probabilities: 0.000 1.000 0.000

Predicted choice: Robot 2

Actual choice: Robot 2

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 6

Participant: 123310

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	0.59274	0.35696	0.29505	0.059274
Robot2	0.56783	0.32455	0.30006	0.056783
Robot3	0.65112	0.41313	0.3031	0.065112

DFT Results:

E\_P: 14.40 40.14 -54.64

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: 7

Participant: 142426

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	0.64294	0.38263	0.3246	0.064294
Robot2	0.6032	0.32883	0.33469	0.06032
Robot3	0.61625	0.33223	0.34564	0.061625

DFT Results:

E\_P: -20.40 19.80 0.49

Choice probabilities: 0.000 1.000 0.000

Predicted choice: Robot 2

Actual choice: Robot 1

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 8

Participant: 175044

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	0.73237	0.44331	0.3623	0.073237
Robot2	0.63258	0.32982	0.36601	0.063258
Robot3	0.59864	0.32401	0.33449	0.059864

DFT Results:

E\_P: -90.92 19.48 71.34

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: 9

Participant: 181700

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	0.76257	0.66706	0.12997
Robot2	1	0.61847	0.64028	0.11443
Robot3	0.84703	0.44494	0.4868	0.084703

DFT Results:

E\_P: -77.32 -75.88 153.10

Choice probabilities: 0.000 0.000 1.000

Predicted choice: Robot 3

Actual choice: Robot 3

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 10

Participant: 214504

Actual Choice: Robot 2

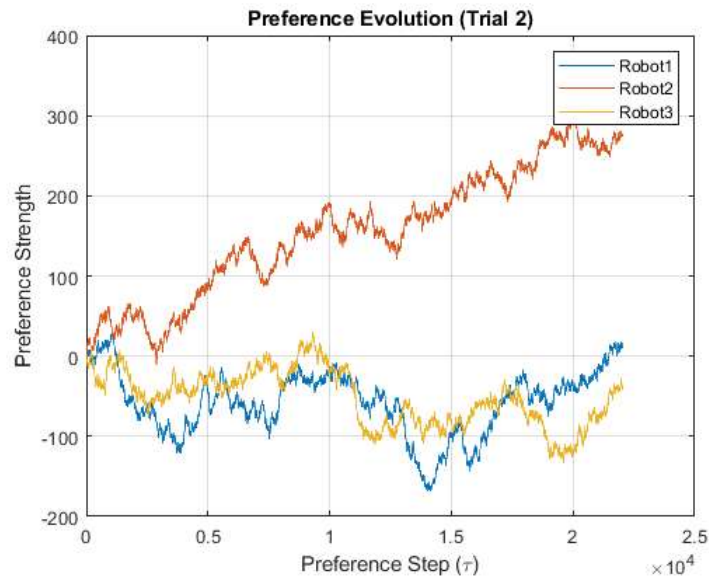
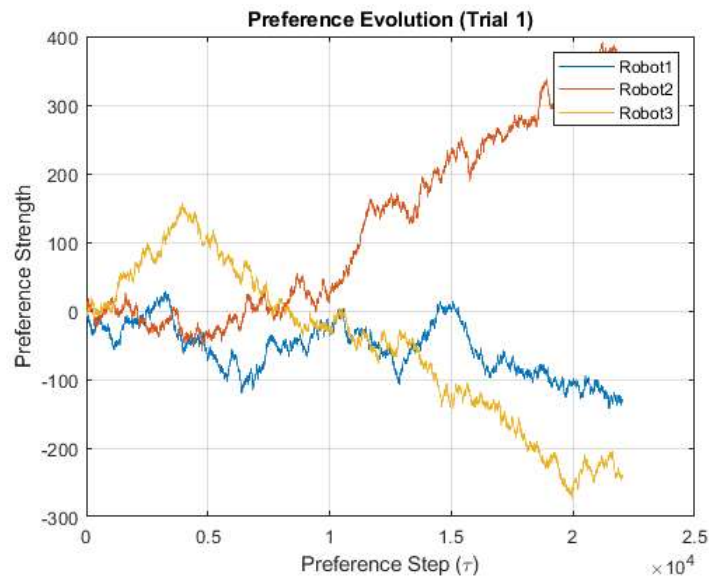
M matrix (alternatives x attributes):

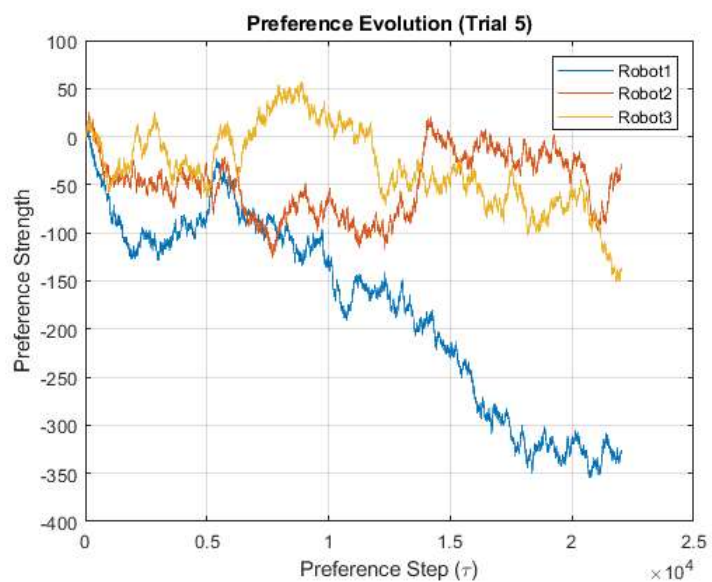
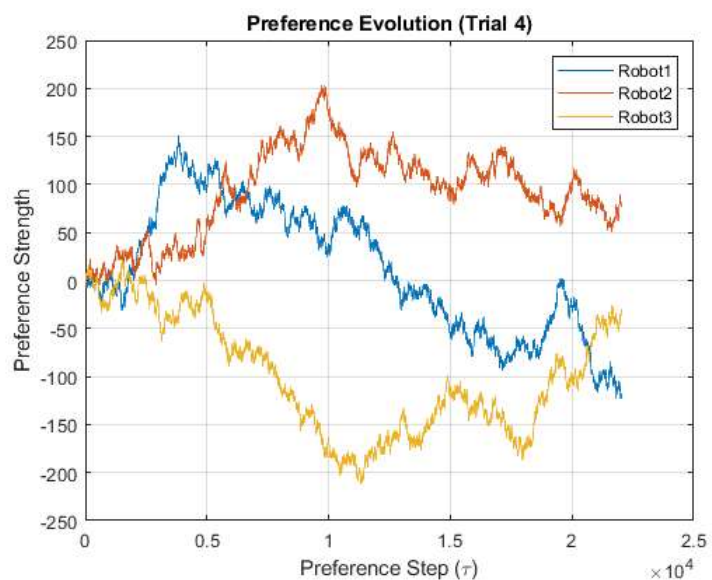
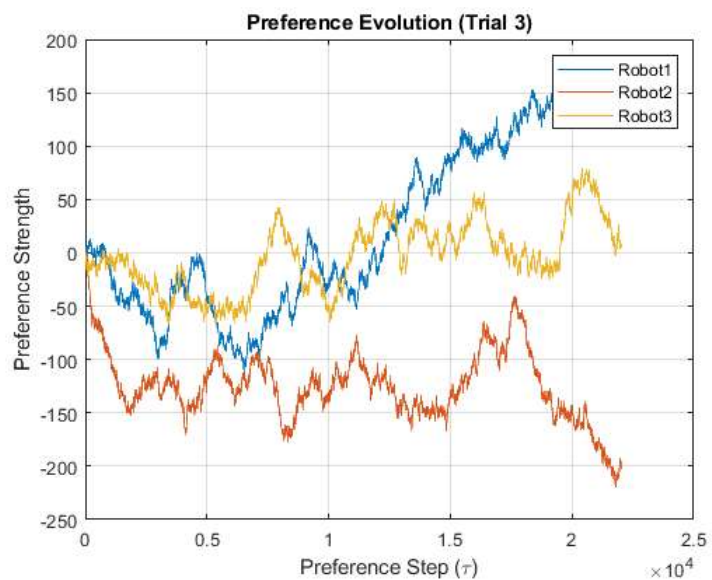
	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	0.59037	0.36202	0.28739	0.059037
Robot2	0.29446	0.17861	0.1453	0.029446
Robot3	0.47225	0.3114	0.20807	0.047225

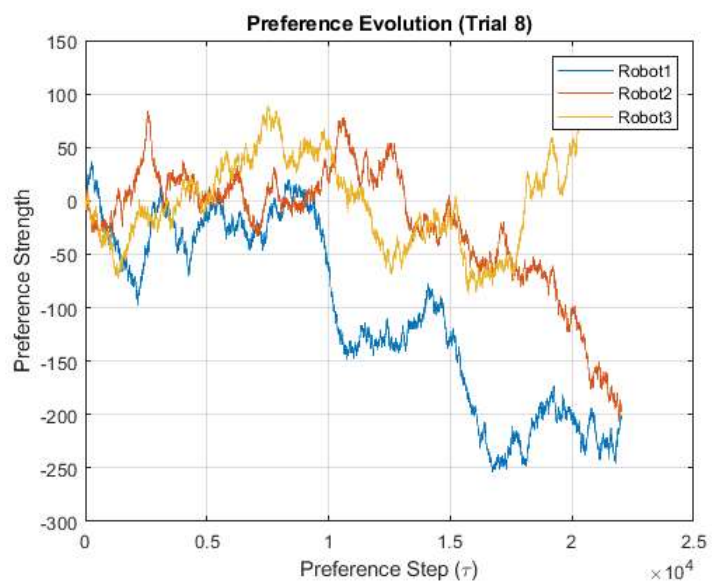
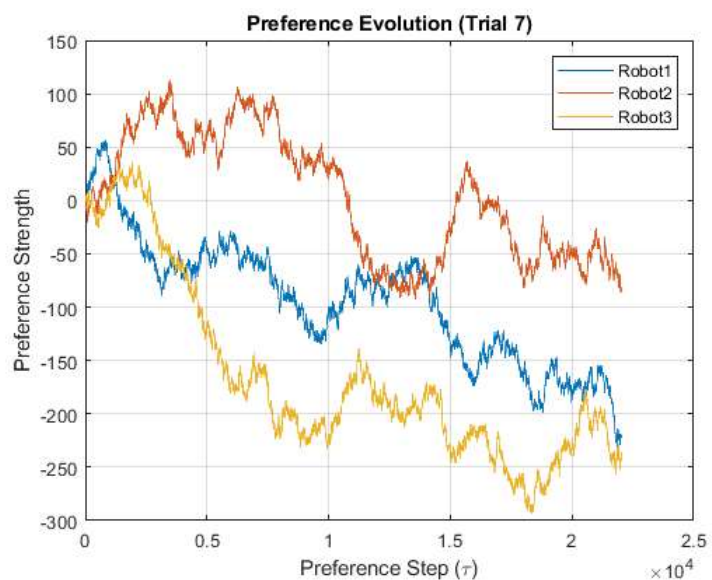
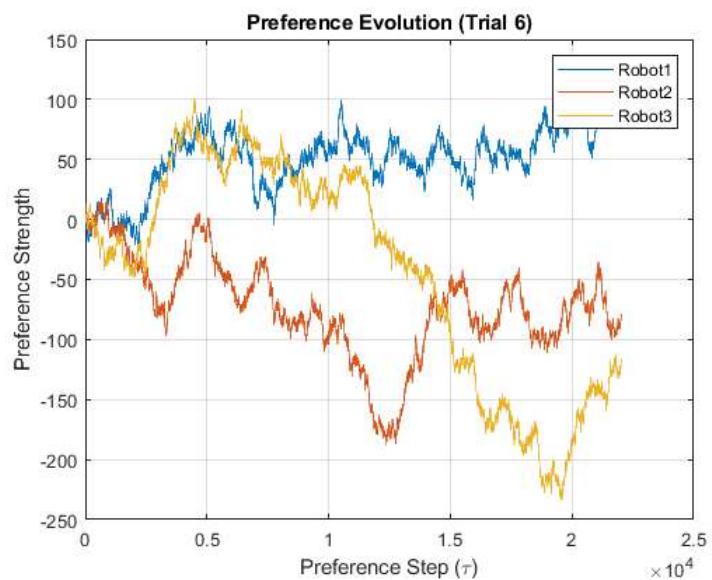
DFT Results:

E\_P: -186.97    206.83    -19.96  
Choice probabilities: 0.000   1.000   0.000  
Predicted choice: Robot 2  
Actual choice: Robot 2

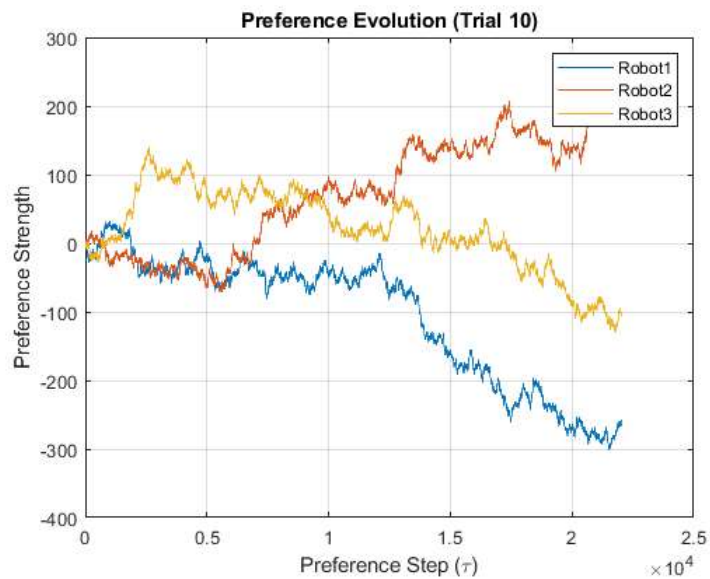
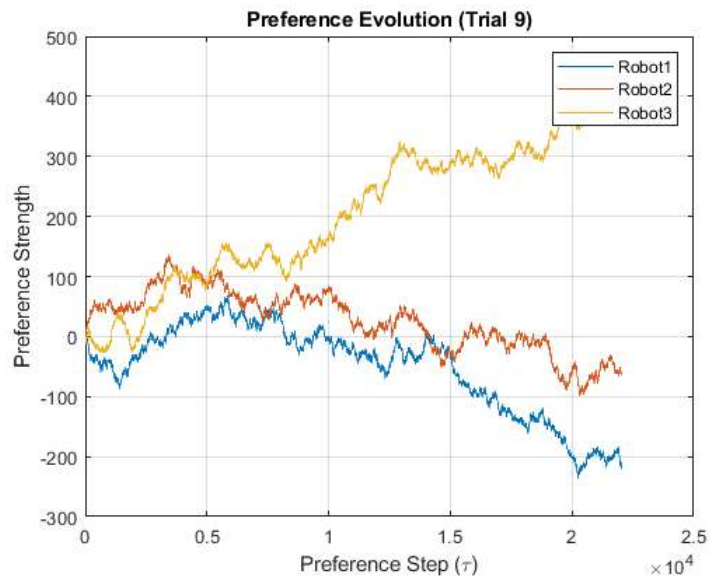
✓ Prediction matches actual choice











## 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*randn();
    warning('Using randomized default parameters');
end
```

Estimated Parameters:

phi1: 3.3786

phi2: 0

tau: 22027.4658

error\_sd: 1

Initial Preferences (from ASCs):

-0.0165   -0.0888   0

