

## 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 Resource Allocation Project with R Integration
% Using Human data to estimate parameters and compare to synthetic data
% The choice predictions from human parameters is used to update survey GUI
% inputs. This helps us validate human-centric choice predictions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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\WarehouseRobot_Pairing_Data\test_pairing_data.csv');
disp('User robot choice data imported successfully.');
```

```
% Extract relevant data (modify based on your CSV structure)
% task_attributes = data{:, {'efficiency', 'speed', 'safety', 'durability', 'skill'}};
```

```
% Extract and organize robot states for all three alternatives
robot_states = struct();
attributes = {'energy', 'pace', 'safety', 'reliability', 'intelligence'};
for i = 1:3
    for attr = attributes
        robot_states.(['robot' num2str(i)]).attr{1} = ...
            robotChoice_Data.(['robot' num2str(i) attr{1}]);
    end
end
```

```
% Extract choice data and other metadata
choices = robotChoice_Data.choice;
participant_ids = robotChoice_Data.participantid;
trial_numbers = robotChoice_Data.trial;
stake_types = robotChoice_Data.staketyp;
time_spent = robotChoice_Data.timespent;
```

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_Resource_Allocation.R';
csvFile = 'G:\My Drive\myResearch\Research Experimentation\Apollo\apollo\data\WarehouseRobot_Pairing_Data\HumanData_Resource_Allocation.csv';
outputDir = 'G:\My Drive\myResearch\Research Experimentation\Apollo\apollo\ResourceAllocation_Output';
```

```
% 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 = exp([
            params.b_energy;
            params.b_pace;
            params.b_safety;
            params.b_reliability;
            params.b_intelligence
        ]);

        % Get initial preferences from ASCs
        initial_P = [
            params.asc_1;
            params.asc_2;
            params.asc_3;
            % 0 % Control alternative1 has 0 initial preference
            % 0 % Control alternative2 has 0 initial preference
        ];

        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]; % Default weights ; 0.1; 0.5
    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)

```

%current_trial = 1; % Analyze first trial (can be looped later)

num_trials = size(robotChoice_Data, 1); % Assuming each row is a trial

```

```

for current_trial = 1:num_trials
    % Extract data for the current trial
    trial_data = robotChoice_Data(current_trial, :);

% Create M matrix from current trial's attributes
M = [
    robotChoice_Data.robot1energy(current_trial), ...
    robotChoice_Data.robot1pace(current_trial), ...
    robotChoice_Data.robot1safety(current_trial), ...
    robotChoice_Data.robot1reliability(current_trial), ...
    robotChoice_Data.robot1intelligence(current_trial);

    robotChoice_Data.robot2energy(current_trial), ...
    robotChoice_Data.robot2pace(current_trial), ...
    robotChoice_Data.robot2safety(current_trial), ...
    robotChoice_Data.robot2reliability(current_trial), ...
    robotChoice_Data.robot2intelligence(current_trial);

    robotChoice_Data.robot3energy(current_trial), ...
    robotChoice_Data.robot3pace(current_trial), ...
    robotChoice_Data.robot3safety(current_trial), ...
    robotChoice_Data.robot3reliability(current_trial), ...
    robotChoice_Data.robot3intelligence(current_trial);

    % 0.1*ones(1,5) % Control alternative1

    % 0.9*ones(1,5) % Control alternative2
];

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

% Calculate DFT dynamics with initial preferences
[E_P, V_P, probs, P_tau] = calculateDFTdynamics(...
    phi1, phi2, tau, error_sd, beta, M, initial_P);

% Display results
disp('=== Current 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)); %,'Control Alt1','Control Alt2'

% Create comparison table
result_table = table(probs, 'VariableNames', {'Probability'}, ...
    'RowNames', {'Robot1','Robot2','Robot3'});
disp('Choice Probability Distribution:'); %,'Control1','Control2'
disp(result_table);

% Display DFT results with prediction
disp('DFT Results:');
disp(['E_P: ', num2str(E_P), '%.2f ']);
disp(['Choice probabilities: ', num2str(probs), '%.3f ']);
[~, predicted_choice] = max(probs); % Get index of highest probability
disp(['Predicted choice: Robot ', num2str(predicted_choice)]);
disp(['Actual choice: Robot ', num2str(choices(current_trial))]);
disp(' ');

% Display match/mismatch
if predicted_choice == choices(current_trial)
    disp('✓ Prediction stayed the same');
else
    disp('X Prediction changed');
end

% Plot preference evolution
figure;
time_steps = 0:size(P_tau, 2)-1;
h = plot(time_steps, P_tau, 'LineWidth', 2);

legend(h, {'Robot 1', 'Robot 2', 'Robot 3'}, 'Location', 'best');
xlabel('Time Step (\tau)');
ylabel('Preference Strength');
title(sprintf('Preference Evolution (Trial %d)', current_trial));

```

```

grid on;

end
%{
%% Step 4: Solve Equilibrium Function
% Ensure DFT outputs match expected dimensions
assert(length(E_P) == 5, 'DFT must return 5 alternatives');
assert(isequal(size(V_P), [5 5]), 'DFT covariance must be 5x5');

% Use DFT outputs for equilibrium calculation
Ep_mins = E_P;      % 4x1 expected preferences from DFT
Varp_mins = V_P;    % 4x4 preference covariance from DFT
% using robot attribute to map to product the robot can build type A vs
% type B. this is made using each robot attribute.
% 5 robot produce 2 things. each trial outputs x_mins as 10x1
x_mins = robot_production_capacity(M); % 10x1 robot state vector could recursive for each neighboring robot

% Call equilibrium solver
solutions = solve_equilibrium(Ep_mins, Varp_mins, x_mins);

% Extract solutions (works with both old and new versions)
if isfield(solutions, 'x')
    % Old version field names
    P_final = solutions.x;
    E_P_eq = solutions.lambda;
    V_P_eq = solutions.mu_vec;
else
    % New version field names
    P_final = solutions.P_final;
    E_P_eq = solutions.E_P_eq;
    V_P_eq = solutions.V_P_eq;
end

% Display results
disp('=== Equilibrium Results ===');
disp(['Final Preferences (P_final): ', num2str(P_final)]);
disp(['Expected Preferences (E_P_eq): ', num2str(E_P_eq)]);
disp(['Preference Variance (V_P_eq diagonal): ', num2str(V_P_eq)]);

%% Step 5: Output Results
disp('Saving results to CSV...');
output_table = table(E_P, V_P, P_final, ...
    'VariableNames', {'ExpectedPreference', 'VariancePreference', 'FinalPreferences'});
writetable(output_table, 'results.csv');
disp('Results saved successfully!');
%}

```

=== Current Trial Analysis ===

Trial: 1

Participant: 1001

Actual Choice: Robot 2

M matrix (alternatives x attributes):

	energy	pace	safety	reliability	intelligence
Robot1	0.65	0.55	0.7	0.6	0.58
Robot2	0.75	0.45	0.8	0.65	0.62
Robot3	0.5	0.6	0.65	0.55	0.5

Choice Probability Distribution:

	Probability
Robot1	0.8042
Robot2	0.12902
Robot3	0.066779

DFT Results:

E\_P: 1.42 -0.41 -1.07

Choice probabilities: 0.804 0.129 0.067

Predicted choice: Robot 1

Actual choice: Robot 2

X Prediction changed

=== Current Trial Analysis ===

Trial: 2

Participant: 1001

Actual Choice: Robot 1

M matrix (alternatives × attributes):

	energy	pace	safety	reliability	intelligence
Robot1	0.55	0.45	0.6	0.5	0.48
Robot2	0.7	0.35	0.75	0.6	0.57
Robot3	0.45	0.55	0.6	0.5	0.45

Choice Probability Distribution:

Probability

Robot1	0.063579
Robot2	0.43985
Robot3	0.49657

DFT Results:

E\_P: -1.31 0.62 0.74

Choice probabilities: 0.064 0.440 0.497

Predicted choice: Robot 3

Actual choice: Robot 1

X Prediction changed

=== Current Trial Analysis ===

Trial: 3

Participant: 1001

Actual Choice: Robot 3

M matrix (alternatives × attributes):

	energy	pace	safety	reliability	intelligence
Robot1	0.6	0.5	0.65	0.55	0.53
Robot2	0.65	0.4	0.7	0.55	0.6
Robot3	0.55	0.65	0.7	0.6	0.55

Choice Probability Distribution:

Probability

Robot1	0.040638
Robot2	0.45837
Robot3	0.50099

DFT Results:

E\_P: -1.63 0.80 0.89

Choice probabilities: 0.041 0.458 0.501

Predicted choice: Robot 3

Actual choice: Robot 3

✓ Prediction stayed the same

=== Current Trial Analysis ===

Trial: 4

Participant: 1002

Actual Choice: Robot 2

M matrix (alternatives × attributes):

	energy	pace	safety	reliability	intelligence
Robot1	0.7	0.6	0.75	0.65	0.63
Robot2	0.8	0.5	0.85	0.7	0.67
Robot3	0.6	0.7	0.75	0.65	0.6

Choice Probability Distribution:

Probability

Robot1	0.044878
Robot2	0.4632
Robot3	0.49192

DFT Results:

E\_P: -1.56 0.77 0.83

Choice probabilities: 0.045 0.463 0.492

Predicted choice: Robot 3

Actual choice: Robot 2

X Prediction changed

=== Current Trial Analysis ===

Trial: 5

Participant: 1002

Actual Choice: Robot 1

M matrix (alternatives × attributes):

	energy	pace	safety	reliability	intelligence
Robot1	0.5	0.4	0.55	0.45	0.43
Robot2	0.6	0.3	0.65	0.5	0.52
Robot3	0.4	0.5	0.55	0.45	0.4

Choice Probability Distribution:

Probability

Robot1	0.036523
Robot2	0.46075
Robot3	0.50273

DFT Results:

E\_P: -1.70 0.84 0.92

Choice probabilities: 0.037 0.461 0.503

Predicted choice: Robot 3

Actual choice: Robot 1

X Prediction changed

=== Current Trial Analysis ===

Trial: 6

Participant: 1002

Actual Choice: Robot 3

M matrix (alternatives × attributes):

	energy	pace	safety	reliability	intelligence
Robot1	0.75	0.65	0.8	0.7	0.68
Robot2	0.85	0.55	0.9	0.75	0.73
Robot3	0.65	0.75	0.8	0.7	0.65

Choice Probability Distribution:

Probability

Robot1	0.041403
Robot2	0.46381
Robot3	0.49479

DFT Results:

E\_P: -1.61 0.80 0.87

Choice probabilities: 0.041 0.464 0.495

Predicted choice: Robot 3

Actual choice: Robot 3

✓ Prediction stayed the same

=== Current Trial Analysis ===

Trial: 7

Participant: 1003

Actual Choice: Robot 2

M matrix (alternatives × attributes):

	energy	pace	safety	reliability	intelligence
Robot1	0.58	0.48	0.63	0.53	0.51
Robot2	0.68	0.38	0.73	0.58	0.61
Robot3	0.48	0.58	0.63	0.53	0.48

Choice Probability Distribution:

Probability

Robot1	0.036978
Robot2	0.45913
Robot3	0.50389

DFT Results:

E\_P: -1.69 0.83 0.92

Choice probabilities: 0.037 0.459 0.504

Predicted choice: Robot 3

Actual choice: Robot 2

X Prediction changed  
 === Current Trial Analysis ===  
 Trial: 8  
 Participant: 1003  
 Actual Choice: Robot 1  
 M matrix (alternatives × attributes):

	energy	pace	safety	reliability	intelligence
Robot1	0.62	0.52	0.67	0.57	0.55
Robot2	0.72	0.42	0.77	0.62	0.65
Robot3	0.52	0.62	0.67	0.57	0.52

Choice Probability Distribution:

	Probability
Robot1	0.036978
Robot2	0.45913
Robot3	0.50389

DFT Results:  
 E\_P: -1.69 0.83 0.92  
 Choice probabilities: 0.037 0.459 0.504  
 Predicted choice: Robot 3  
 Actual choice: Robot 1

X Prediction changed  
 === Current Trial Analysis ===  
 Trial: 9  
 Participant: 1003  
 Actual Choice: Robot 3  
 M matrix (alternatives × attributes):

	energy	pace	safety	reliability	intelligence
Robot1	0.67	0.57	0.72	0.62	0.6
Robot2	0.77	0.47	0.82	0.67	0.7
Robot3	0.57	0.67	0.72	0.62	0.57

Choice Probability Distribution:

	Probability
Robot1	0.036978
Robot2	0.45913
Robot3	0.50389

DFT Results:  
 E\_P: -1.69 0.83 0.92  
 Choice probabilities: 0.037 0.459 0.504  
 Predicted choice: Robot 3  
 Actual choice: Robot 3

✓ Prediction stayed the same  
 === Current Trial Analysis ===  
 Trial: 10  
 Participant: 1004  
 Actual Choice: Robot 2  
 M matrix (alternatives × attributes):

	energy	pace	safety	reliability	intelligence
Robot1	0.53	0.43	0.58	0.48	0.46
Robot2	0.63	0.33	0.68	0.53	0.56
Robot3	0.43	0.53	0.58	0.48	0.43

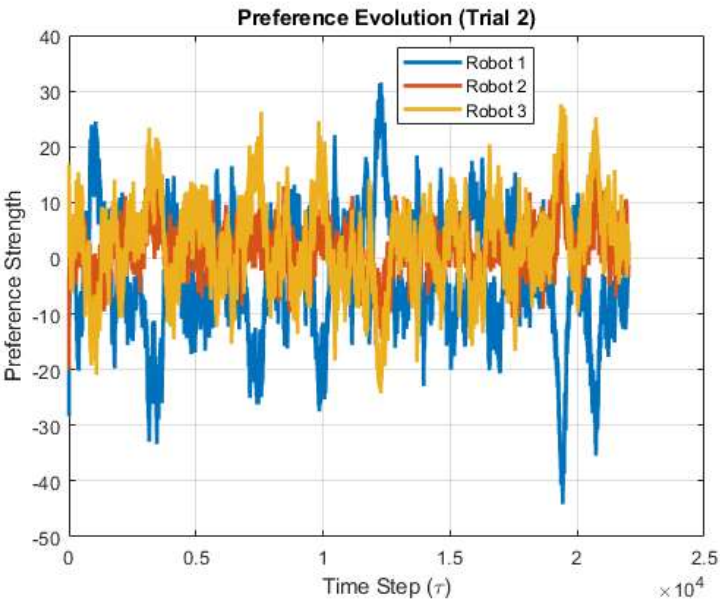
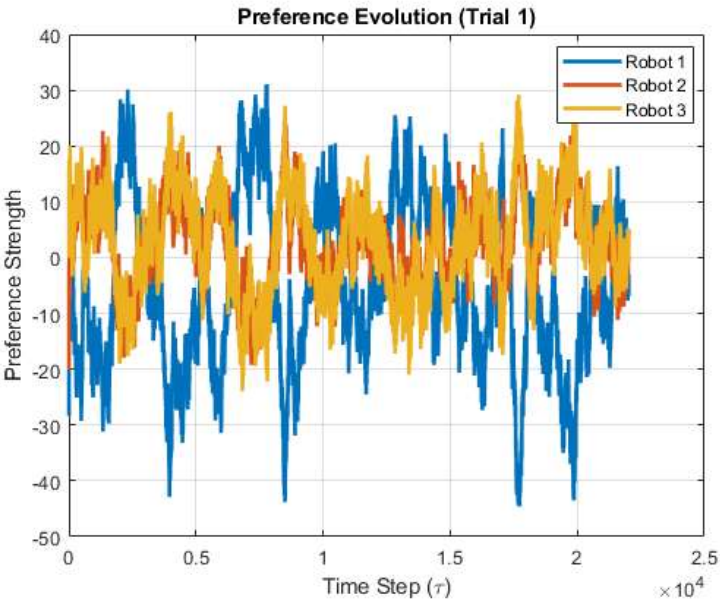
Choice Probability Distribution:

	Probability
Robot1	0.036978
Robot2	0.45913
Robot3	0.50389

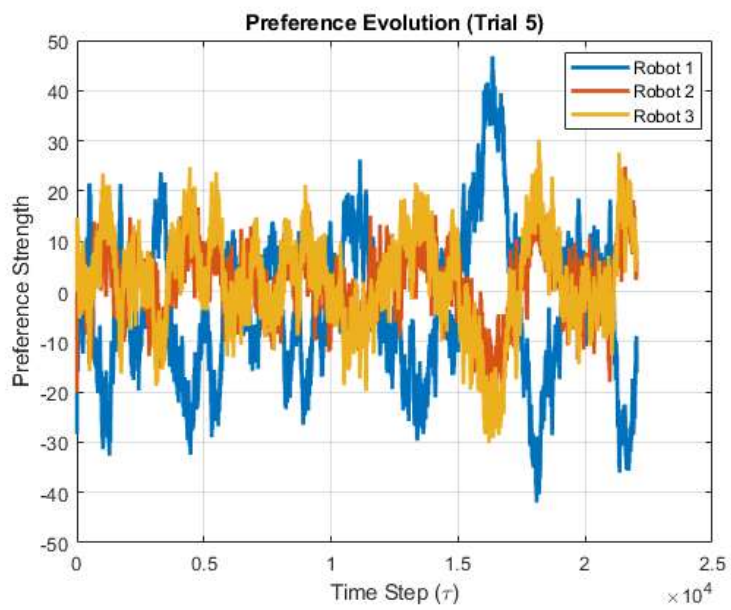
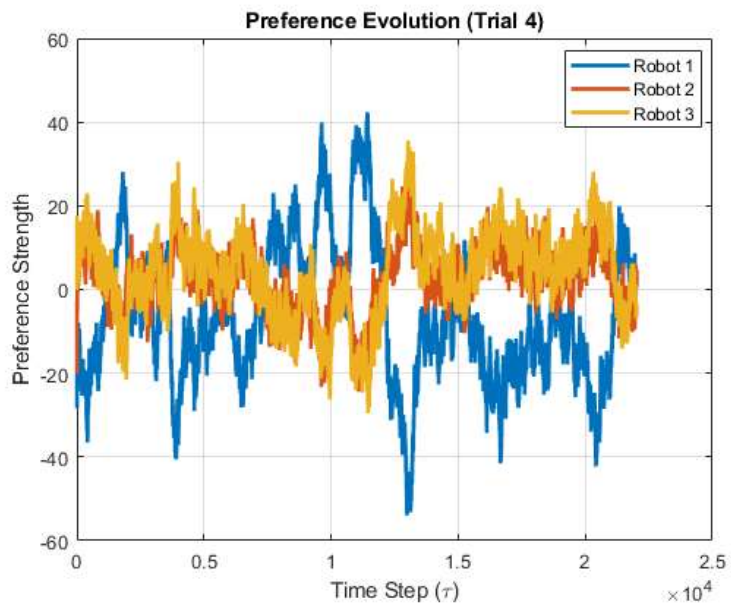
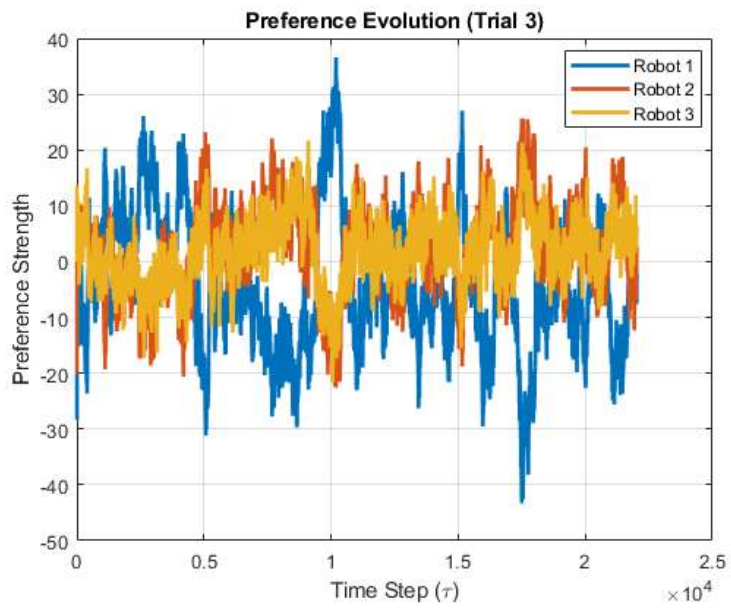
DFT Results:  
 E\_P: -1.69 0.83 0.92  
 Choice probabilities: 0.037 0.459 0.504  
 Predicted choice: Robot 3

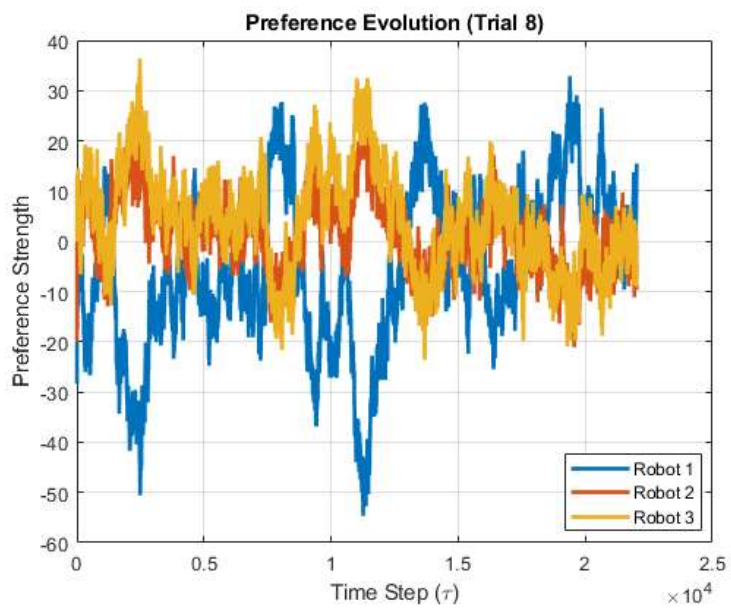
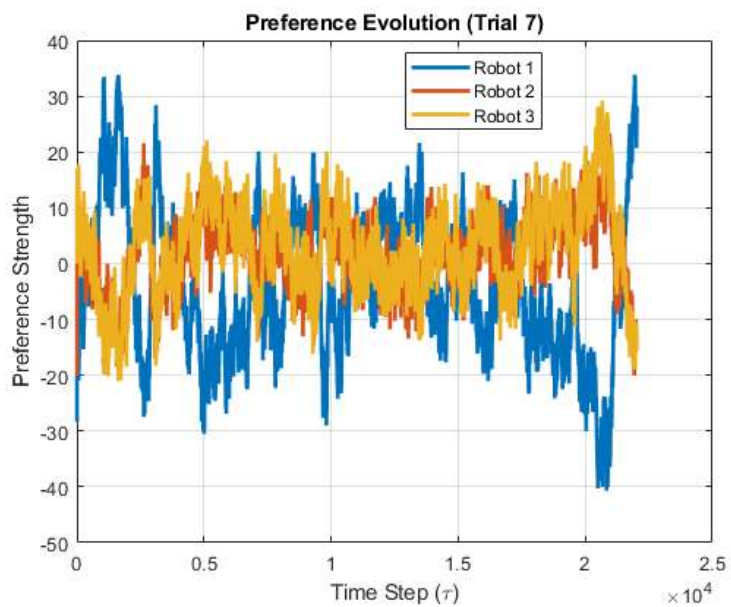
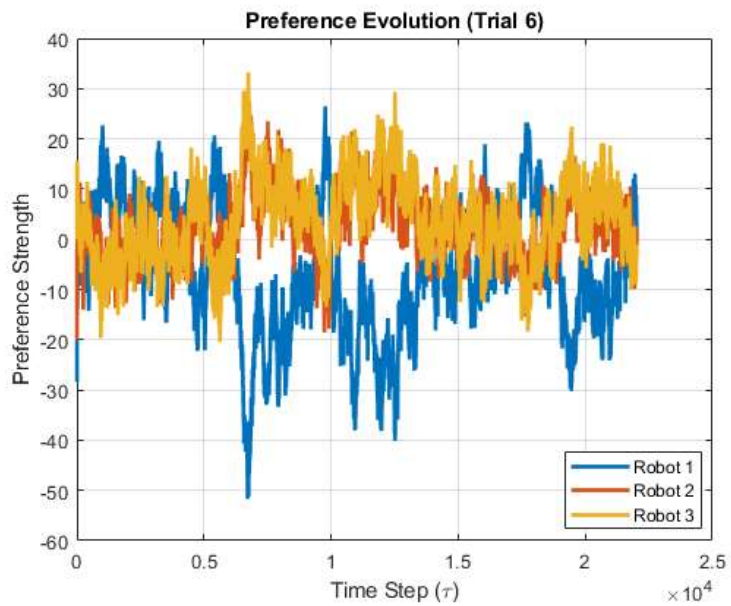
Actual choice: Robot 2

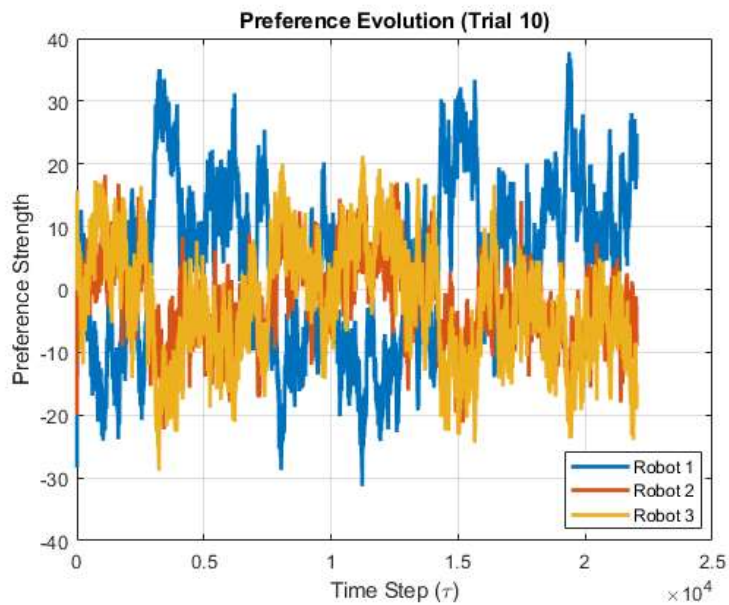
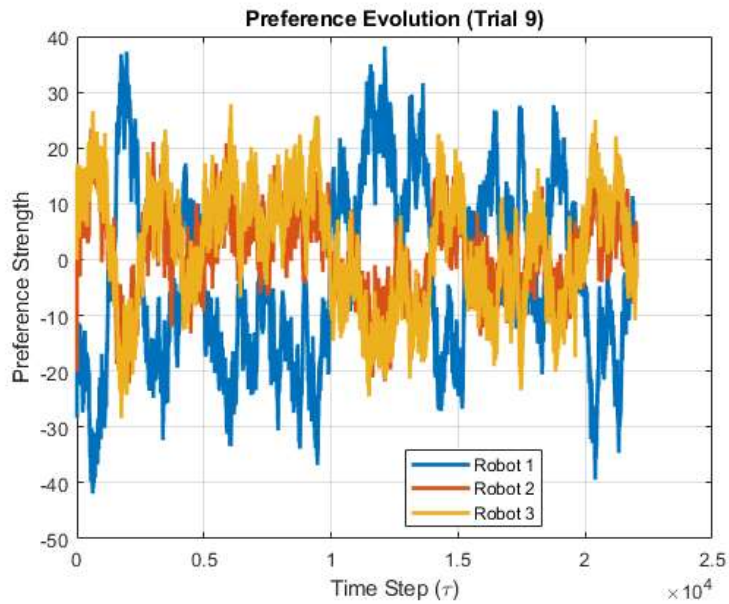
X Prediction changed











## 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;
    phi2 = 0.8;
    tau = 10;
    error_sd = 0.1;
    warning('Using default parameters');
end
```

Estimated Parameters:  
 phi1: 35.5485  
 phi2: 0.1852  
 tau: 22027.4658  
 error\_sd: 1  
 Initial Preferences (from ASCs):

-28.3579   -20.0494   0