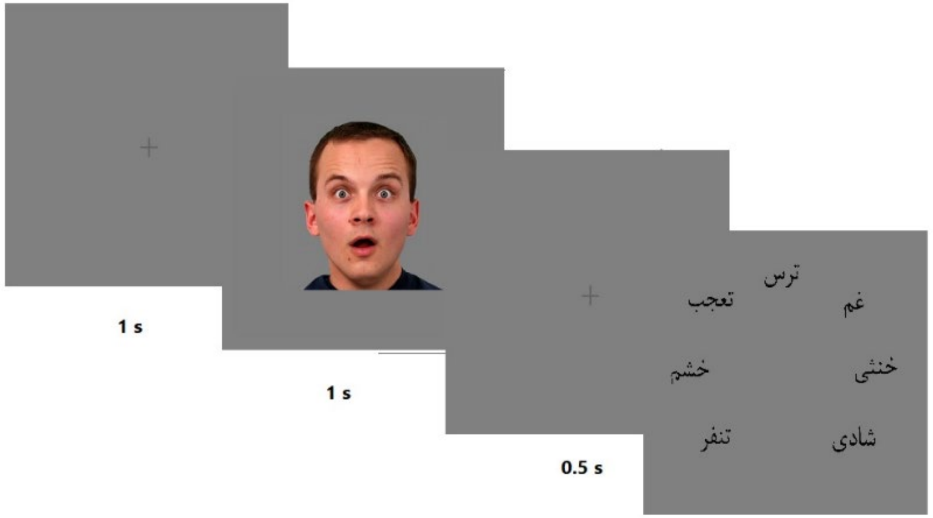
MATLAB for Brain and Cognitive Psychology (Full Experiment)

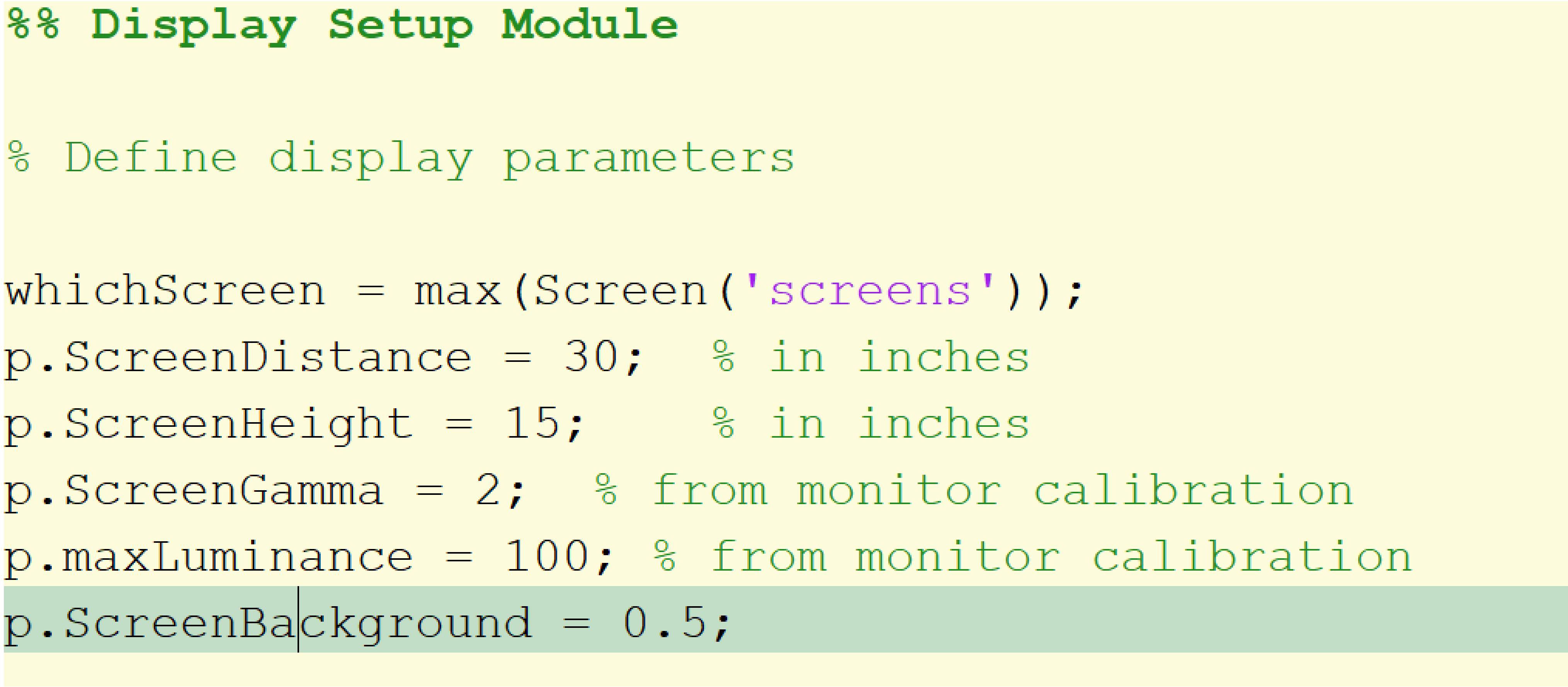
Presented by:
Ehsan Rezayat, Ph.D.
Faculty of Psychology and Education, University of Tehran.
Institute for Research in Fundamental Sciences (IPM), School of Cognitive Sciences,
emails: rezayat@ut.ac.ir, rezayat@ipm.ir, erezayat.er@gmail.com

Experiment

- Subject
- Sessions
- Conditions
- Trial
- Performance
- Reaction time
- Accuracy

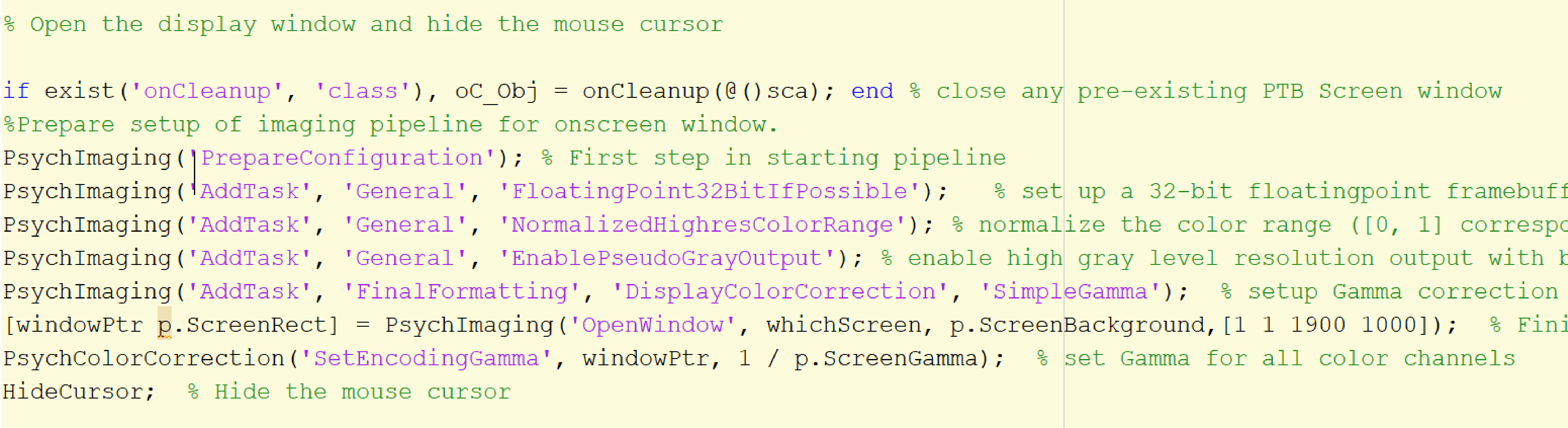


Display Setup Module



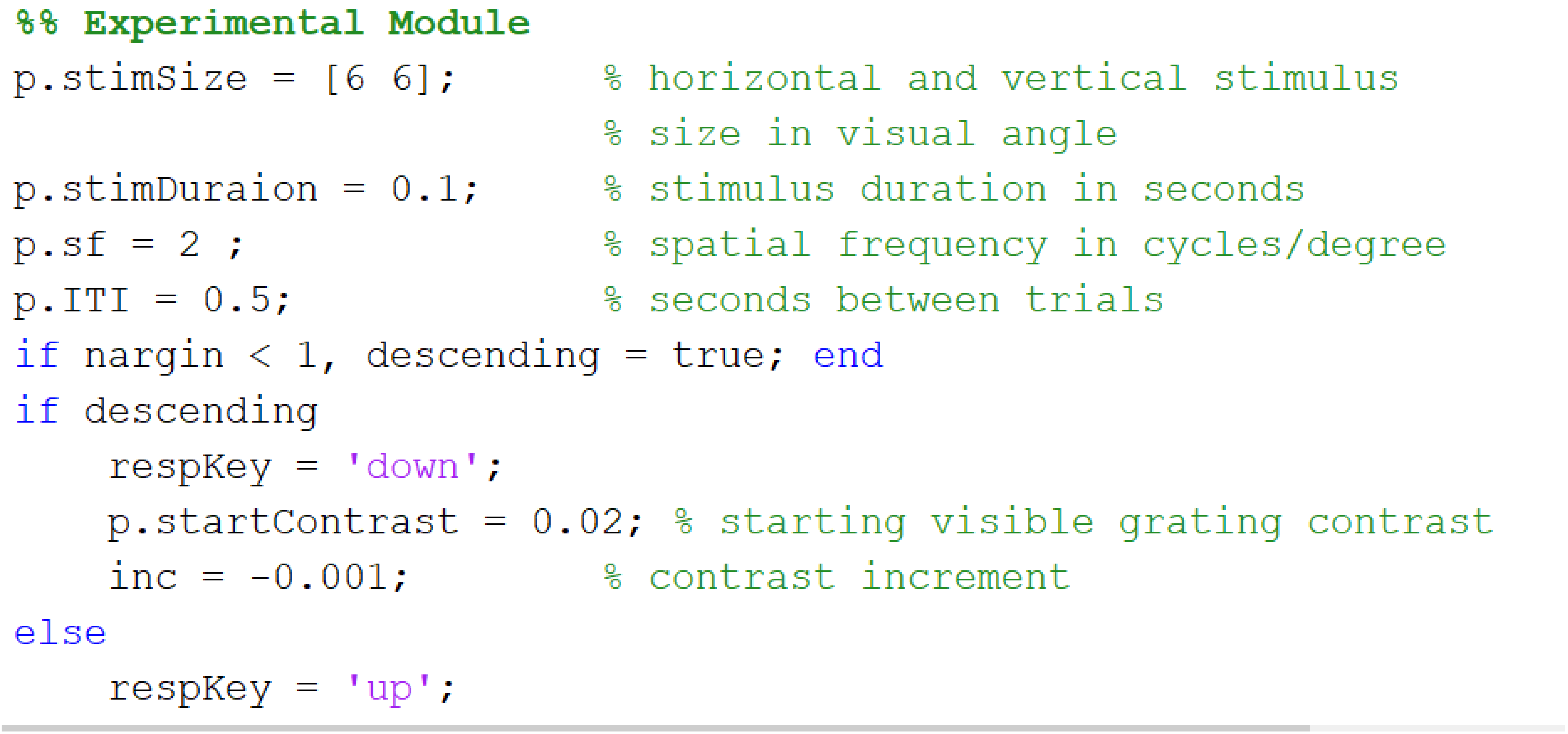
```matlab
%% Display Setup Module

% Define display parameters

whichScreen = max(Screen('screens'));
p.ScreenDistance = 30;  % in inches
p.ScreenHeight = 15;    % in inches
p.ScreenGamma = 2;  % from monitor calibration
p.maxLuminance = 100; % from monitor calibration
p.ScreenBackground = 0.5;
```

Display Setup Module



```
% Open the display window and hide the mouse cursor

if exist('onCleanup', 'class'), oC_Obj = onCleanup(@()sca); end % close any pre-existing PTB Screen window
%Prepare setup of imaging pipeline for onscreen window.
PsychImaging('PrepareConfiguration'); % First step in starting pipeline
PsychImaging('AddTask', 'General', 'FloatingPoint32BitIfPossible');   % set up a 32-bit floatingpoint framebuf
PsychImaging('AddTask', 'General', 'NormalizedHighresColorRange'); % normalize the color range ([0, 1] correspo
PsychImaging('AddTask', 'General', 'EnablePseudoGrayOutput'); % enable high gray level resolution output with b
PsychImaging('AddTask', 'FinalFormatting', 'DisplayColorCorrection', 'SimpleGamma');  % setup Gamma correction
[windowPtr p.ScreenRect] = PsychImaging('OpenWindow', whichScreen, p.ScreenBackground,[1 1 1900 1000]);  % Fini
PsychColorCorrection('SetEncodingGamma', windowPtr, 1 / p.ScreenGamma);  % set Gamma for all color channels
HideCursor;  % Hide the mouse cursor
```

Experimental Module

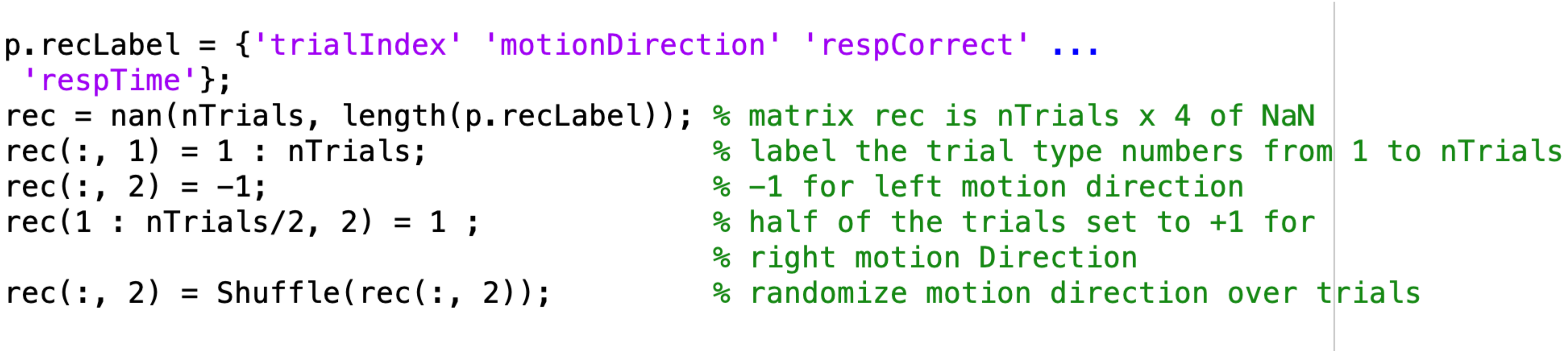


```
%% Experimental Module
p.stimSize = [6 6];      % horizontal and vertical stimulus
                         % size in visual angle
p.stimDuraion = 0.1;     % stimulus duration in seconds
p.sf = 2 ;               % spatial frequency in cycles/degree
p.ITI = 0.5;             % seconds between trials
if nargin < 1, descending = true; end
if descending
    respKey = 'down';
    p.startContrast = 0.02; % starting visible grating contrast
    inc = -0.001;        % contrast increment
else
    respKey = 'up';
```

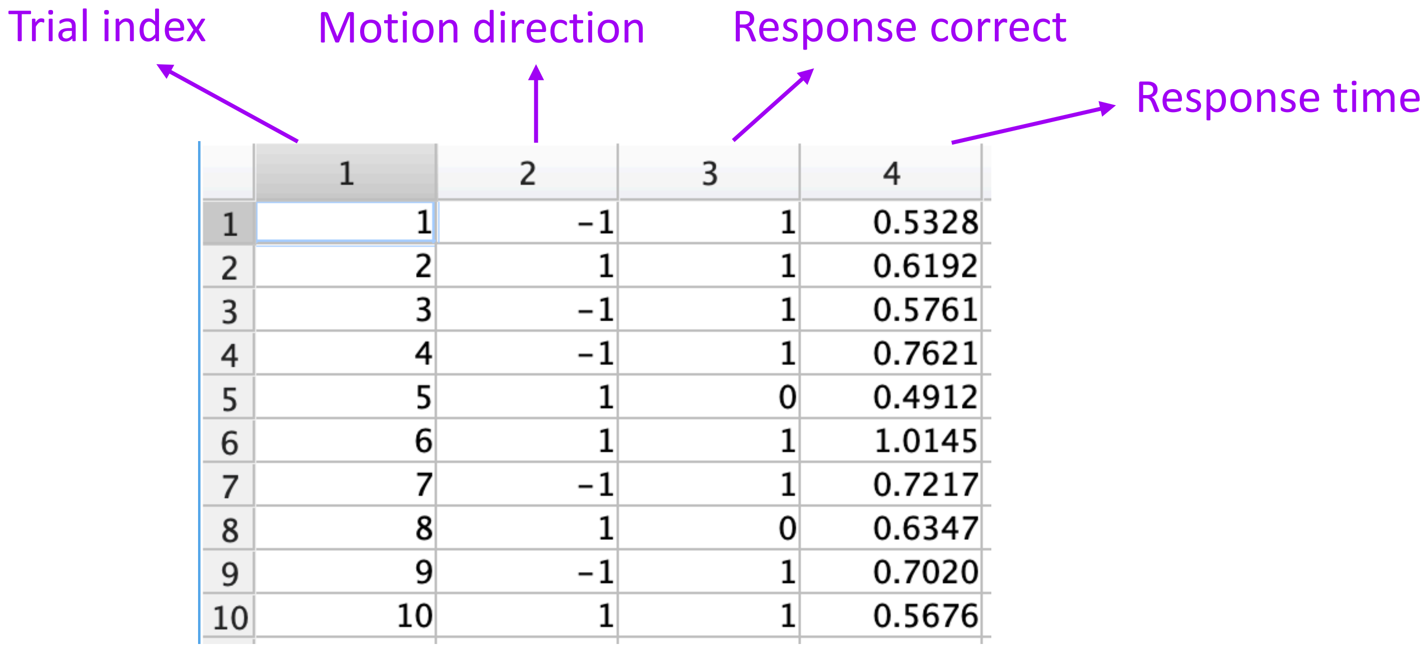
Specify the stimulus

```
p.stimSize = [8 6];       % horizontal and vertical stimulus size
                          % in visual angle
p.stimDuration = 0.250;   % stimulus duration in seconds
p.ISI = 0.5;              % duration between response and next trial onset
p.contrast = 0.2;         % grating contrast
p.tf = 4;                 % drifting temporal frequency in Hz
p.sf = 1;                 % spatial frequency in cycles/degree

% grating contrast
% drifting temporal frequency in Hz
% spatial frequency in cycles/degree
% Compute stimulus parameters
ppd = pi/180 * p.ScreenDistance / p.ScreenHeight * ...
p.ScreenRect(4); % pixels per degree
nFrames = round(p.stimDuration * p.ScreenFrameRate);
% stimulus frames
m = 2 * round(p.stimSize * ppd / 2);% horizontal and vertical
                                    % stimulus size in pixels
sf = p.sf / ppd; % cycles per pixel
phasePerFrame = 360 * p.tf / p.ScreenFrameRate; % phase drift per frame
fixRect = CenterRect([0 0 1 1] * 8, p.ScreenRect); % 8 x 8 fixation
params = [0 sf p.contrast 0];
% parameters for DrawTexture: initial phase, spatial
% frequency, contrast, 0
% procedural sinewavegrating on GPU sets up a texture for
% the sine wave
tex = CreateProceduralSineGrating(windowPtr, m(1), m(2), ...
[1 1 1 0] * 0.5, [], 0.5);
```

Initialize a table to set up experimental conditions



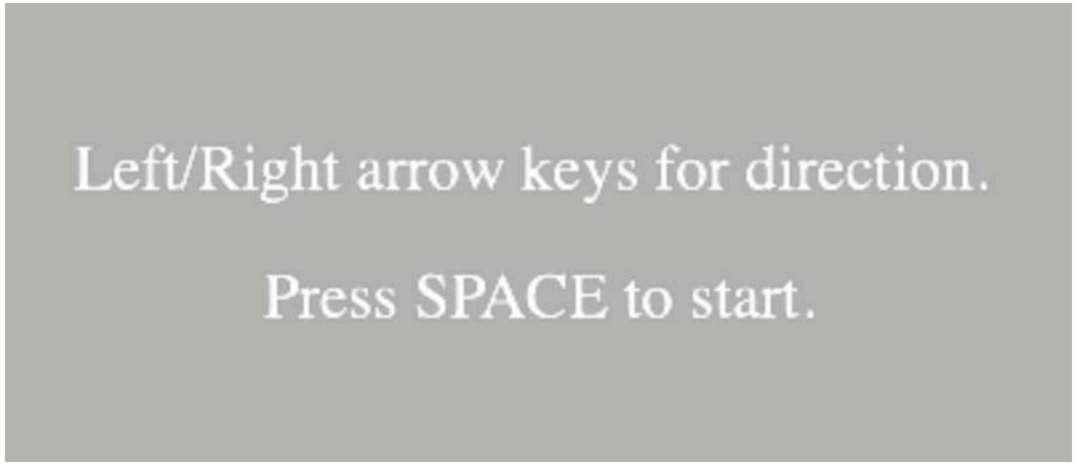
```
p.recLabel = {'trialIndex' 'motionDirection' 'respCorrect' ...
 'respTime'};
rec = nan(nTrials, length(p.recLabel)); % matrix rec is nTrials x 4 of NaN
rec(:, 1) = 1 : nTrials;                % label the trial type numbers from 1 to nTrials
rec(:, 2) = −1;                         % −1 for left motion direction
rec(1 : nTrials/2, 2) = 1 ;             % half of the trials set to +1 for
                                        % right motion Direction
rec(:, 2) = Shuffle(rec(:, 2));         % randomize motion direction over trials
```



	1	2	3	4
1	1	-1	1	0.5328
2	2	1	1	0.6192
3	3	-1	1	0.5761
4	4	-1	1	0.7621
5	5	1	0	0.4912
6	6	1	1	1.0145
7	7	-1	1	0.7217
8	8	1	0	0.6347
9	9	-1	1	0.7020
10	10	1	1	0.5676

Start experiment with instructions

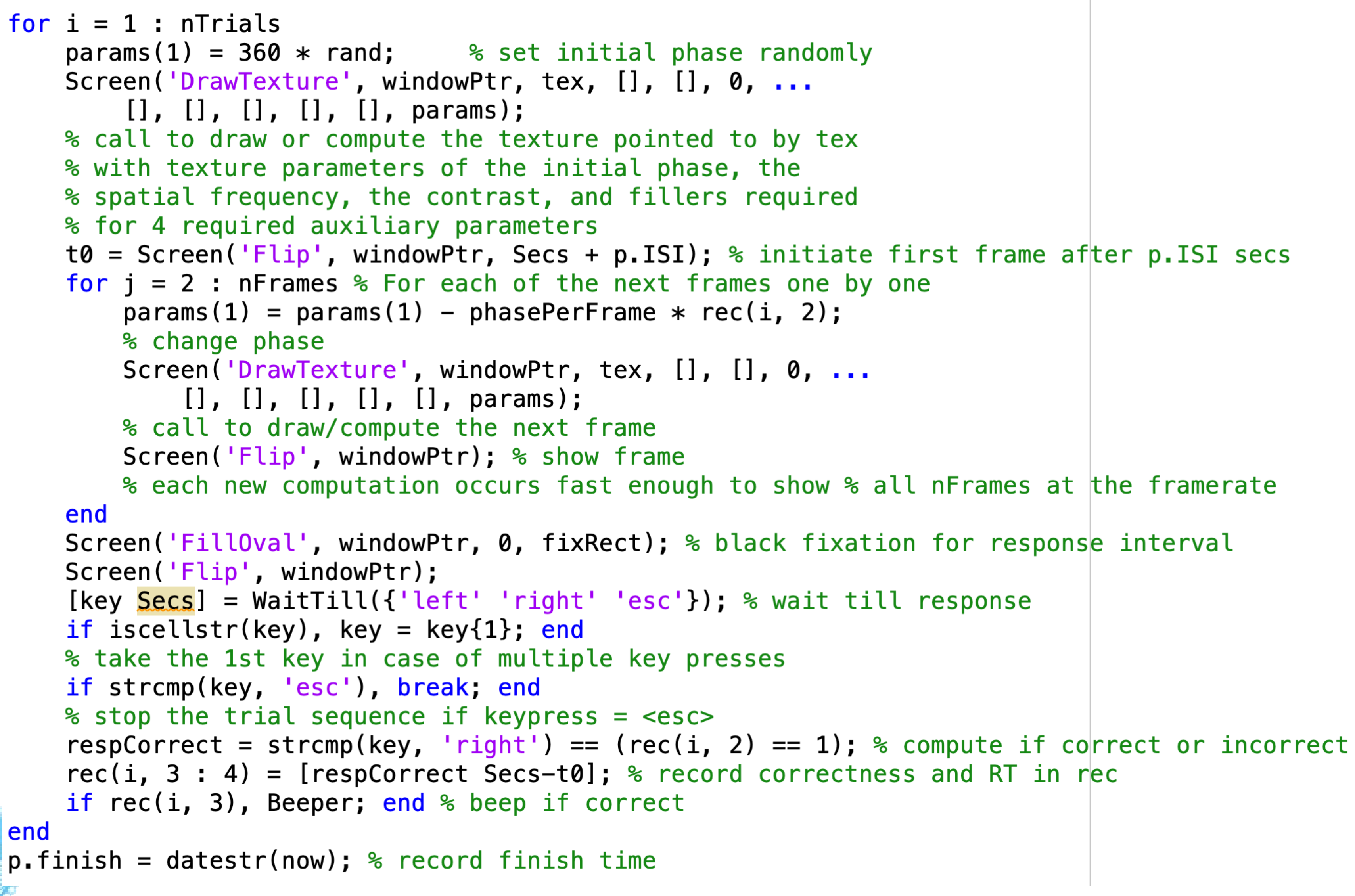
```
str = sprintf('Left/Right arrow keys for direction.\n\n Press SPACE to start.');

DrawFormattedText(windowPtr, str, 'center', 'center', 1);
% Draw instruction text string centered in window
Screen('Flip', windowPtr);
% flip the text image into active buffer
WaitTill('space'); % wait till space bar is pressed
Screen('FillOval', windowPtr, 0, fixRect);
% create fixation box as black (0)
Secs = Screen('Flip', windowPtr);
% flip the fixation image into active buffer
p.start = datestr(now); % record start time
```



Left/Right arrow keys for direction.

Press SPACE to start.

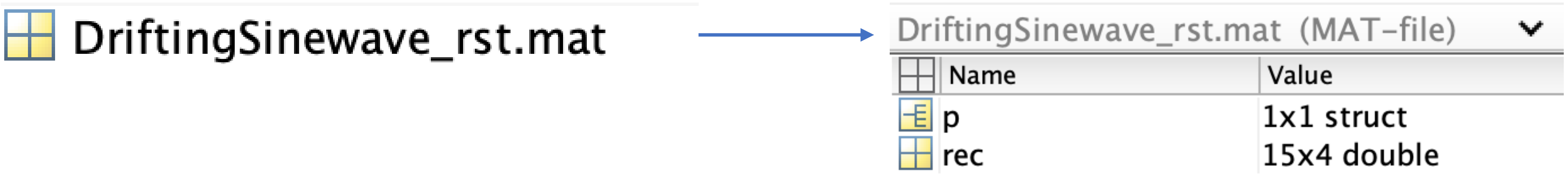
Run nTrials trials



```
for i = 1 : nTrials
    params(1) = 360 * rand;     % set initial phase randomly
    Screen('DrawTexture', windowPtr, tex, [], [], 0, ...
        [], [], [], [], [], params);
    % call to draw or compute the texture pointed to by tex
    % with texture parameters of the initial phase, the
    % spatial frequency, the contrast, and fillers required
    % for 4 required auxiliary parameters
    t0 = Screen('Flip', windowPtr, Secs + p.ISI); % initiate first frame after p.ISI secs
    for j = 2 : nFrames % For each of the next frames one by one
        params(1) = params(1) - phasePerFrame * rec(i, 2);
        % change phase
        Screen('DrawTexture', windowPtr, tex, [], [], 0, ...
            [], [], [], [], [], params);
        % call to draw/compute the next frame
        Screen('Flip', windowPtr); % show frame
        % each new computation occurs fast enough to show % all nFrames at the framerate
    end
    Screen('FillOval', windowPtr, 0, fixRect); % black fixation for response interval
    Screen('Flip', windowPtr);
    [key Secs] = WaitTill({'left' 'right' 'esc'}); % wait till response
    if iscellstr(key), key = key{1}; end
    % take the 1st key in case of multiple key presses
    if strcmp(key, 'esc'), break; end
    % stop the trial sequence if keypress = <esc>
    respCorrect = strcmp(key, 'right') == (rec(i, 2) == 1); % compute if correct or incorrect
    rec(i, 3 : 4) = [respCorrect Secs-t0]; % record correctness and RT in rec
    if rec(i, 3), Beeper; end % beep if correct
end
p.finish = datestr(now); % record finish time
```

Save Results

```
save DriftingSinewave_rst.mat rec p; % save the result
```



DriftingSinewave_rst.mat

DriftingSinewave_rst.mat (MAT–file)

Name	Value
p	1x1 struct
rec	15x4 double

Assignment #11

Run exemplar experiment **Method of Constant Stimuli** on your system and get more data.