دانشکده روانشناسی و علوم تربیتی

IPM
پژوهشگاه دانش‌های بنیادی

MATLAB for Brain and Cognitive Psychology (Exemplar Experiments)

Presented by:

Ehsan Rezayat, Ph.D.

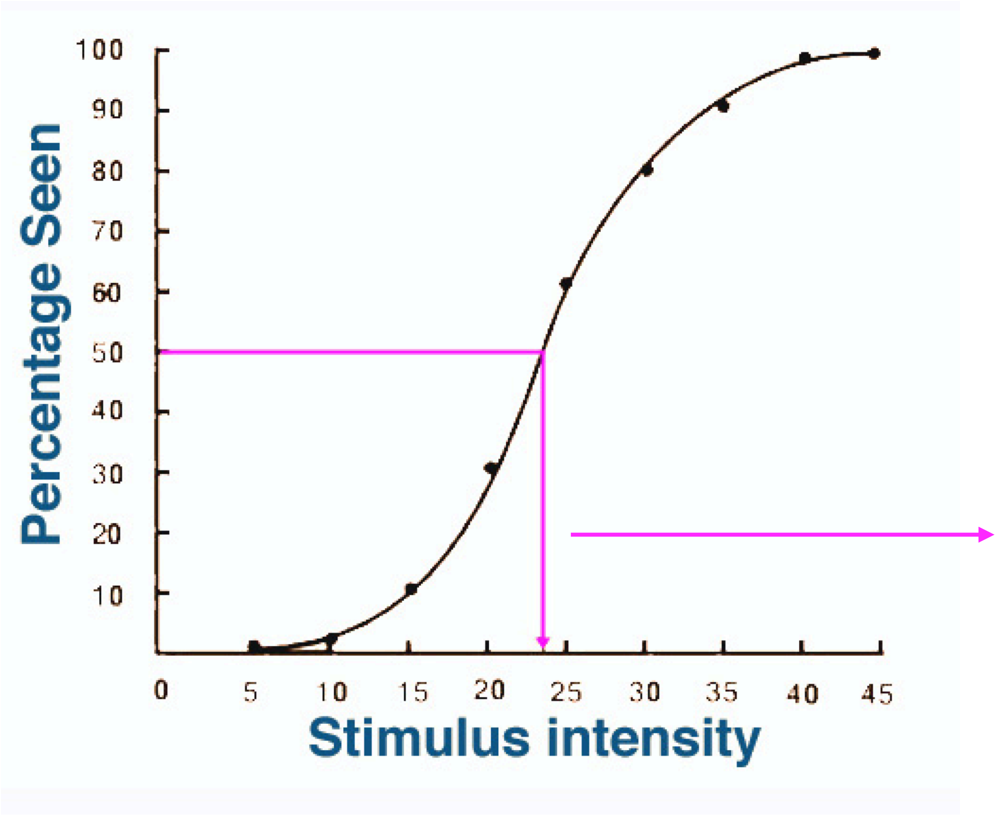
Faculty of Psychology and Education, University of Tehran.

Institute for Research in Fundamental Sciences (IPM), School of Cognitive Sciences,

emails: rezayat@ut.ac.ir, rezayat@ipm.ir, erezayat.er@gmail.com

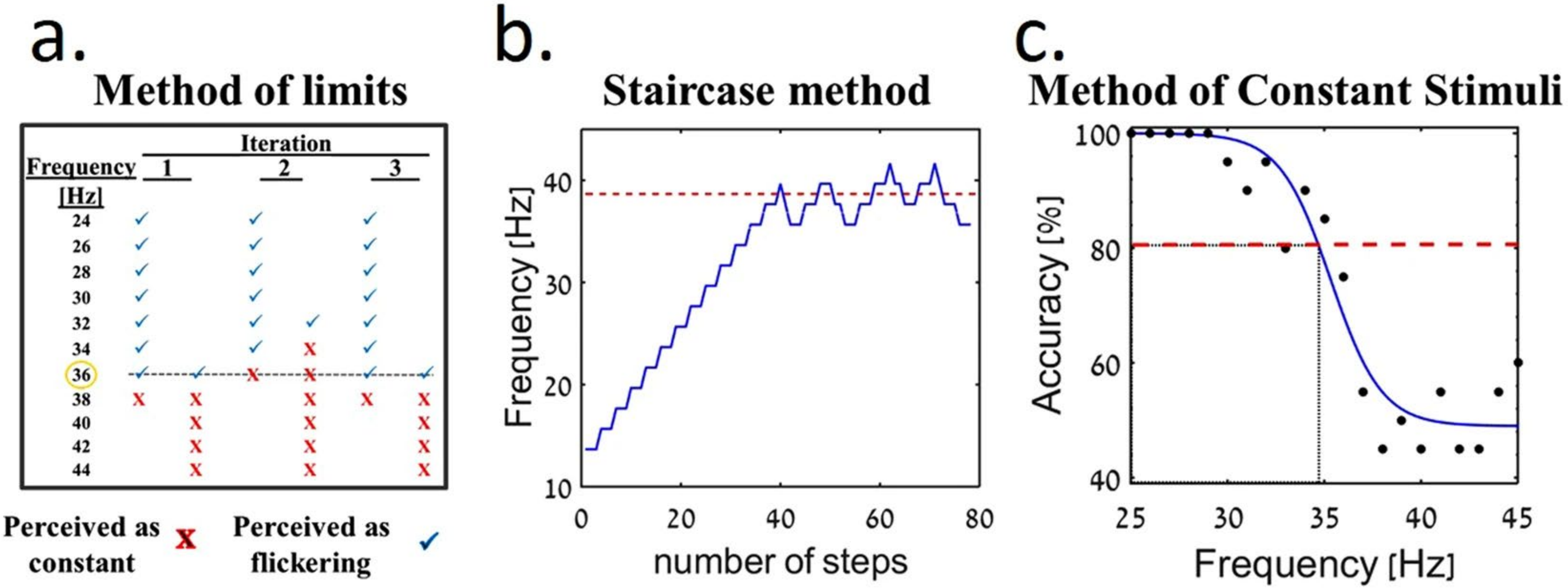
Threshold Demo

- The inflection point of the sigmoid function or the point at which the function reaches the middle between the chance level and 100% is usually taken as sensory threshold.



Threshold

Sensitivity and Threshold



a. Method of limits

Frequency [Hz]	Iteration 1		Iteration 2		Iteration 3	
24	✓		✓		✓	
26	✓		✓		✓	
28	✓		✓		✓	
30	✓		✓		✓	
32	✓		✓	✓	✓	
34	✓		✓	X	✓	
36	✓	✓	X	X	✓	✓
38	X	X		X	X	X
40		X		X		X
42		X		X		X
44		X		X		X

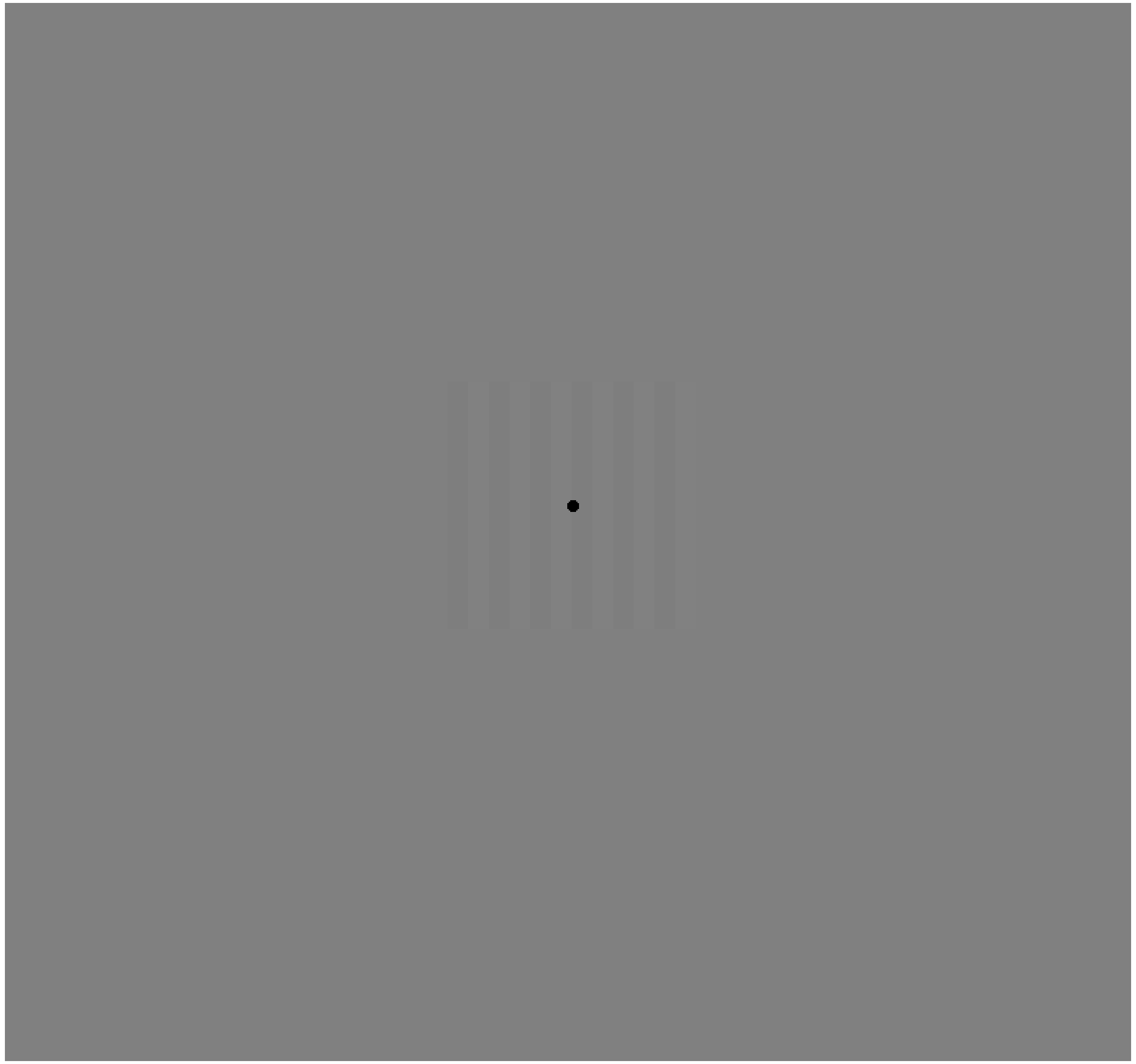
Perceived as constant X Perceived as flickering ✓

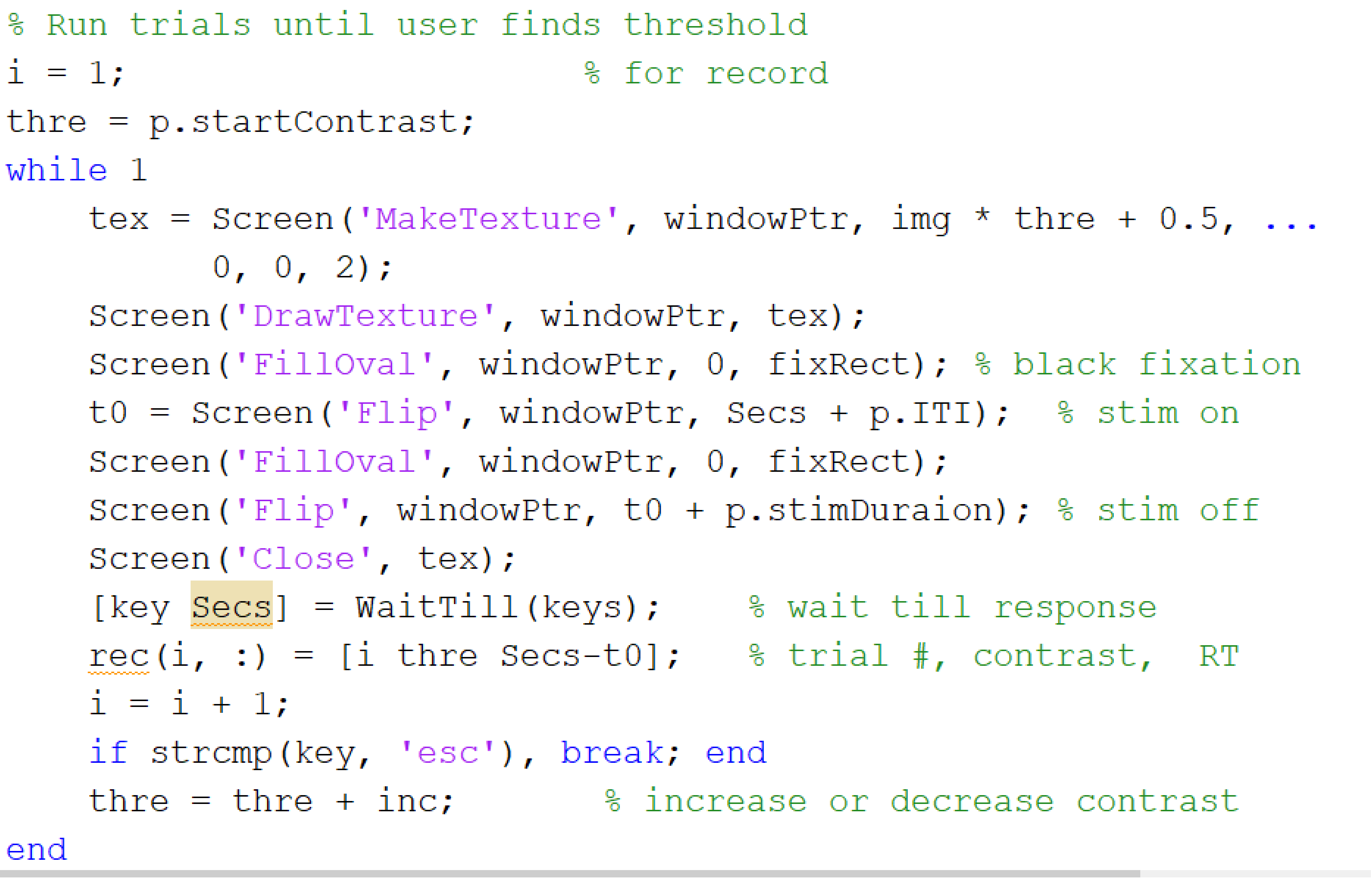
b. Staircase method

c. Method of Constant Stimuli

Eisen-Enosh, A., Farah, N., Burgansky-Eliash, Z., Polat, U., & Mandel, Y. (2017). Evaluation of critical flicker-fusion frequency measurement methods for the investigation of visual temporal resolution. *Scientific reports*, *7*(1), 15621.

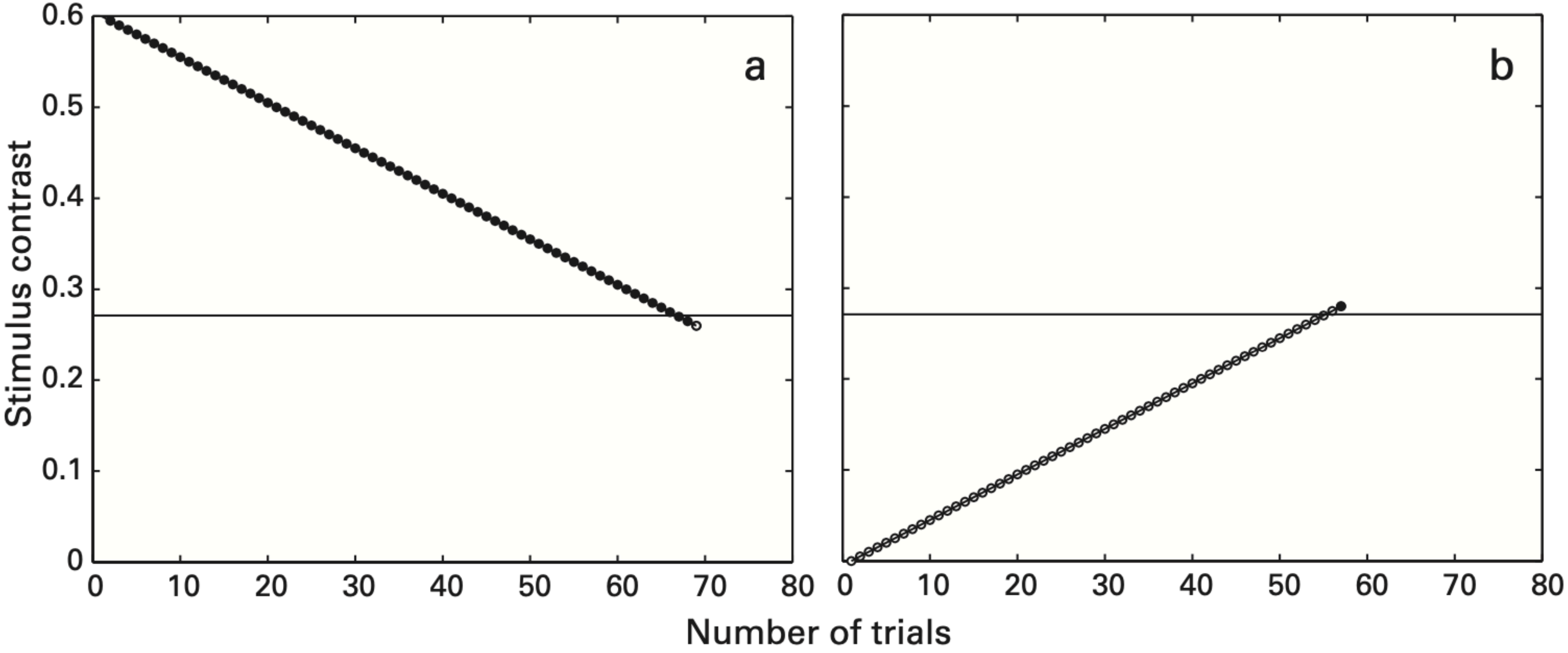
- Method of Limit



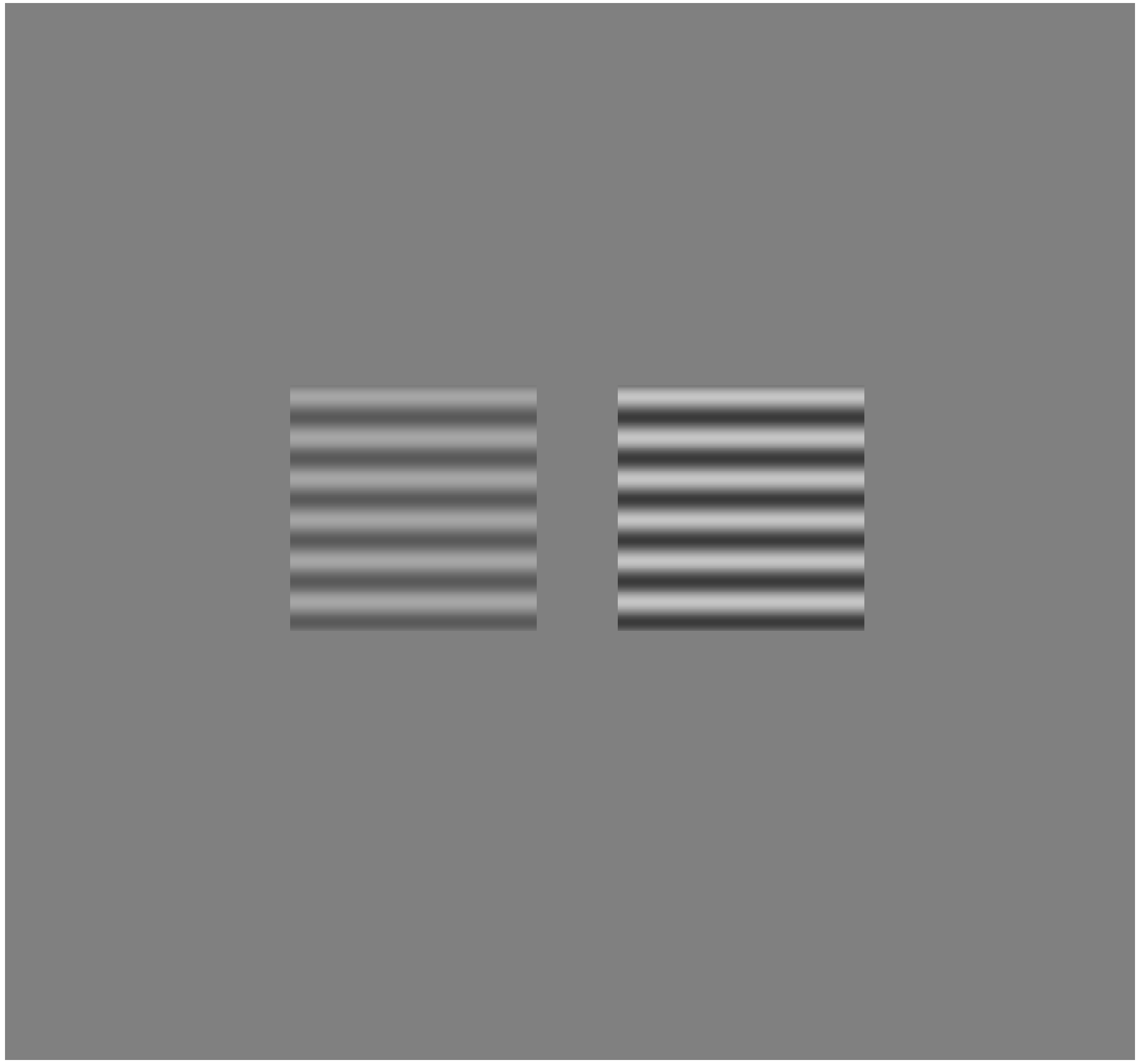


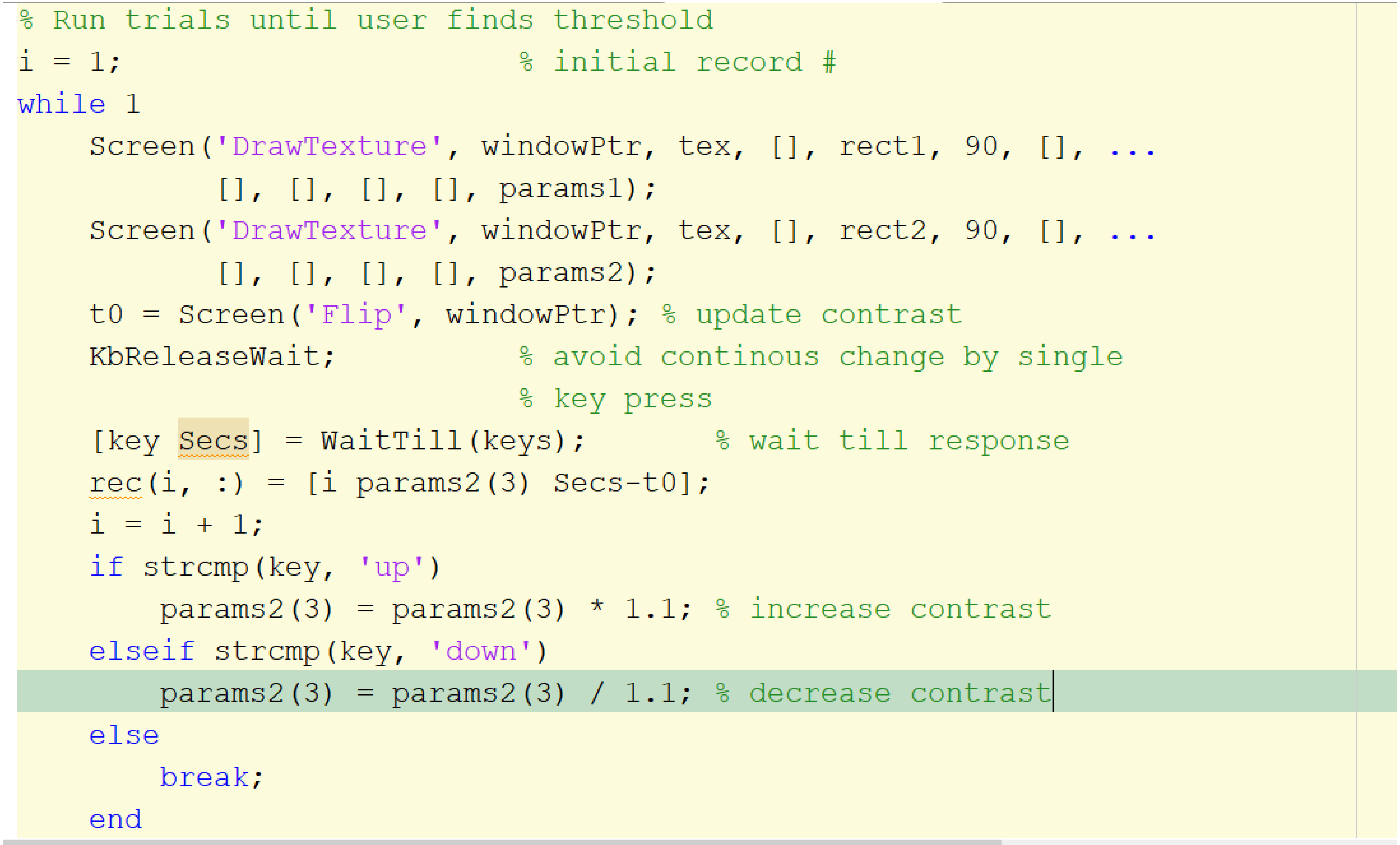
```
% Run trials until user finds threshold
i = 1;                      % for record
thre = p.startContrast;
while 1
    tex = Screen('MakeTexture', windowPtr, img * thre + 0.5, ...
          0, 0, 2);
    Screen('DrawTexture', windowPtr, tex);
    Screen('FillOval', windowPtr, 0, fixRect); % black fixation
    t0 = Screen('Flip', windowPtr, Secs + p.ITI);  % stim on
    Screen('FillOval', windowPtr, 0, fixRect);
    Screen('Flip', windowPtr, t0 + p.stimDuraion); % stim off
    Screen('Close', tex);
    [key Secs] = WaitTill(keys);    % wait till response
    rec(i, :) = [i thre Secs-t0];   % trial #, contrast,  RT
    i = i + 1;
    if strcmp(key, 'esc'), break; end
    thre = thre + inc;       % increase or decrease contrast
end
```

Method of Limits



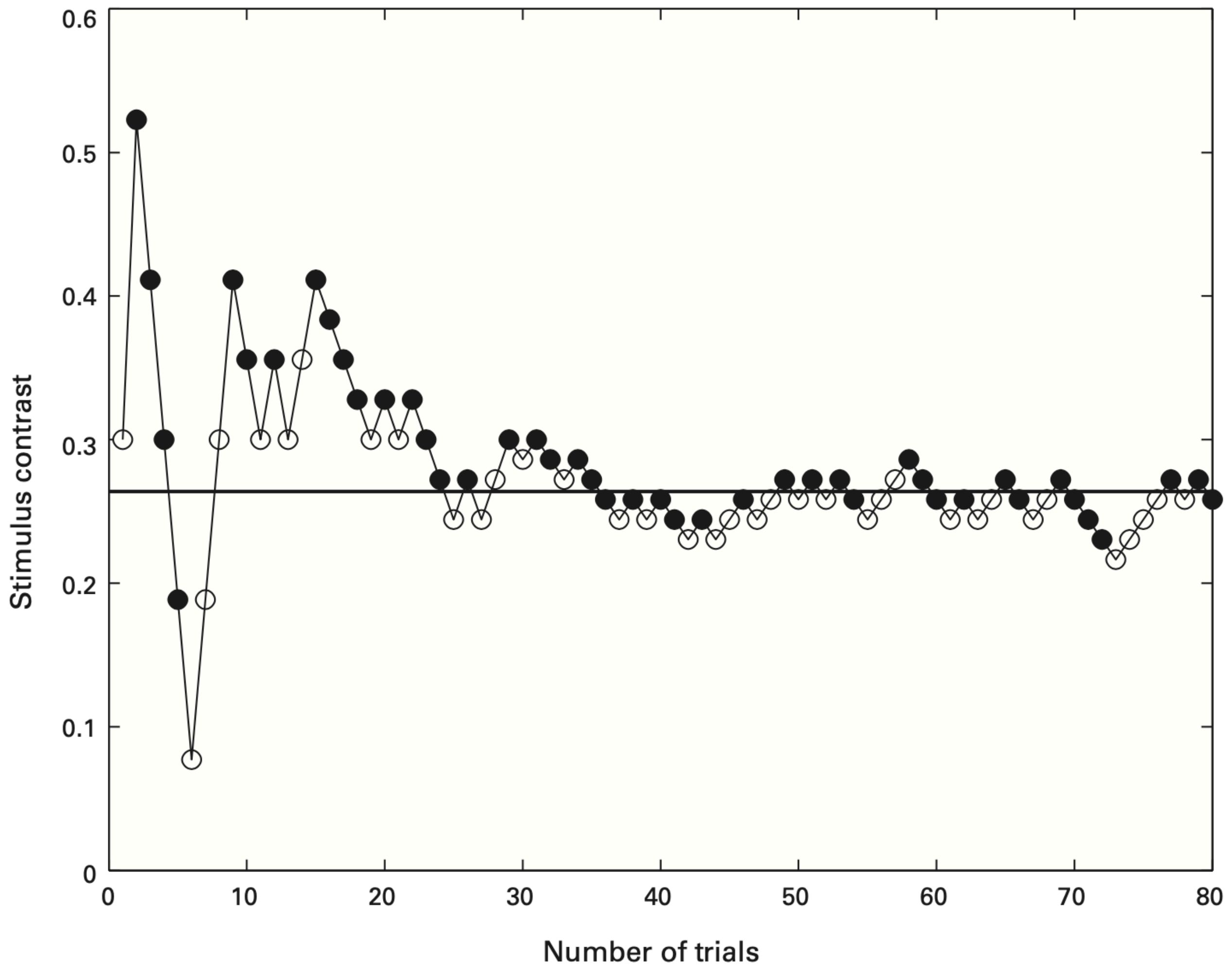
- Method of adjustment



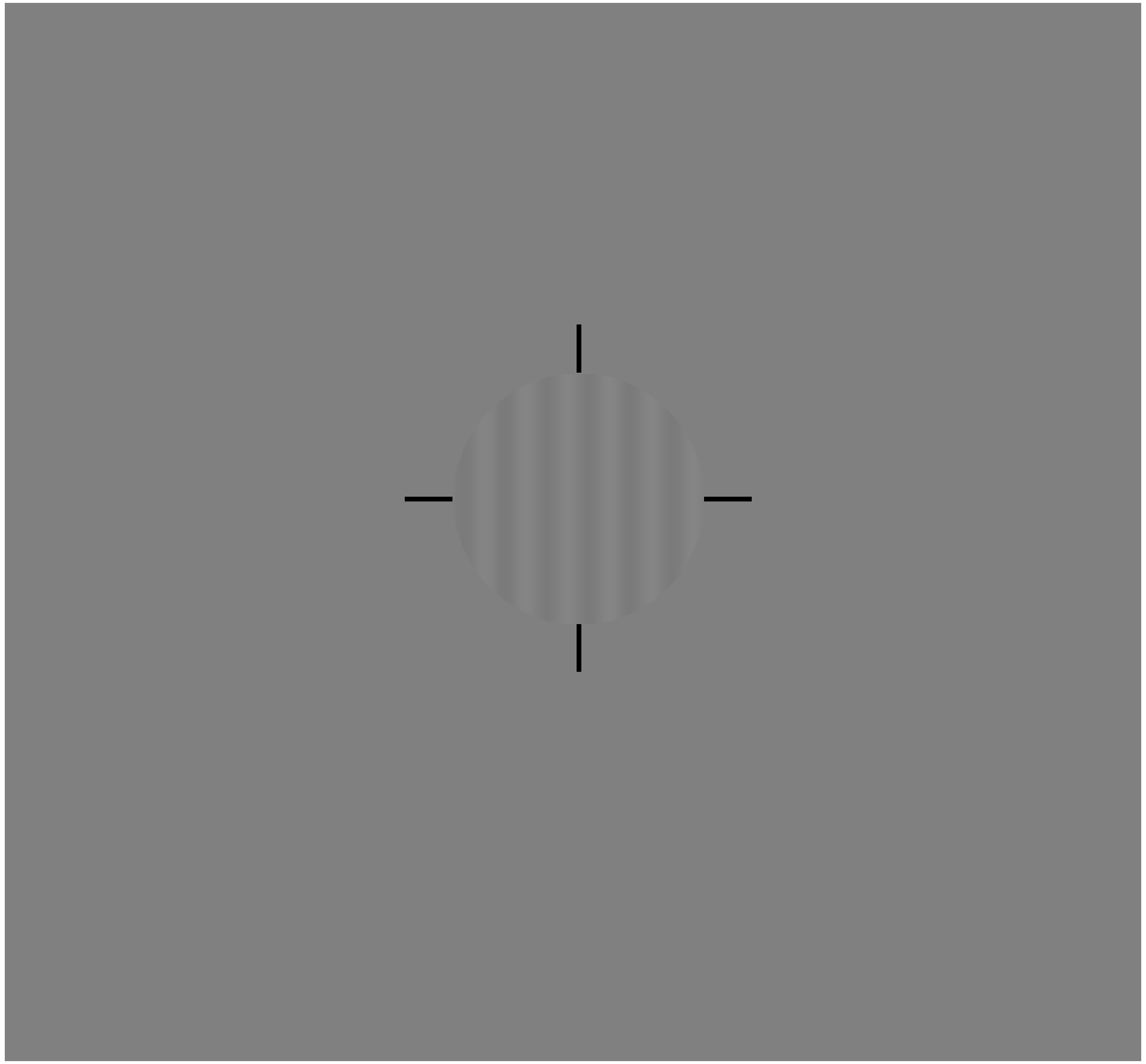


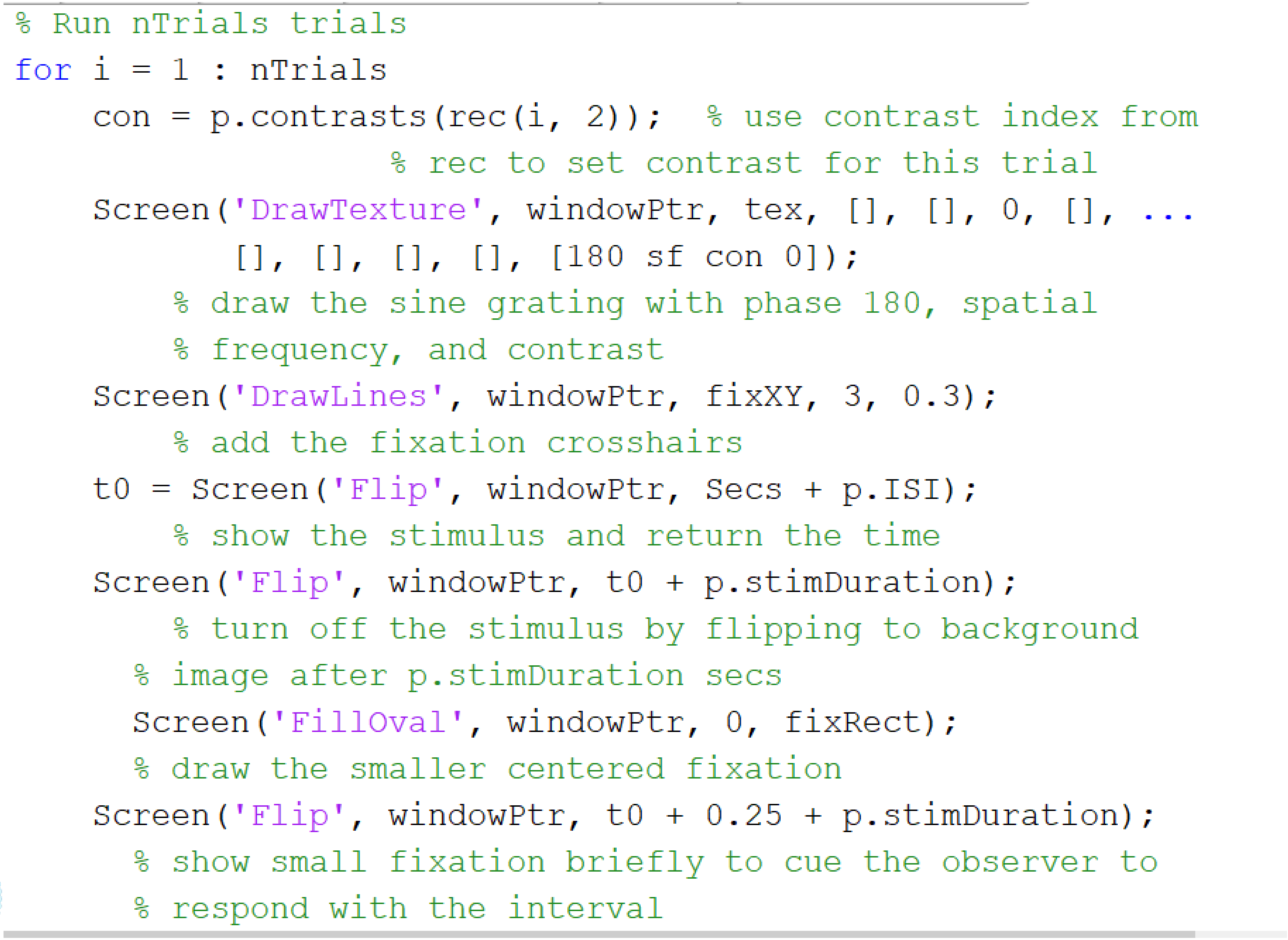
```
% Run trials until user finds threshold
i = 1;                      % initial record #
while 1
    Screen('DrawTexture', windowPtr, tex, [], rect1, 90, [], ...
           [], [], [], [], params1);
    Screen('DrawTexture', windowPtr, tex, [], rect2, 90, [], ...
           [], [], [], [], params2);
    t0 = Screen('Flip', windowPtr); % update contrast
    KbReleaseWait;          % avoid continous change by single
                            % key press
    [key Secs] = WaitTill(keys);       % wait till response
    rec(i, :) = [i params2(3) Secs-t0];
    i = i + 1;
    if strcmp(key, 'up')
        params2(3) = params2(3) * 1.1; % increase contrast
    elseif strcmp(key, 'down')
        params2(3) = params2(3) / 1.1; % decrease contrast
    else
        break;
    end
```

Method of Adjustment



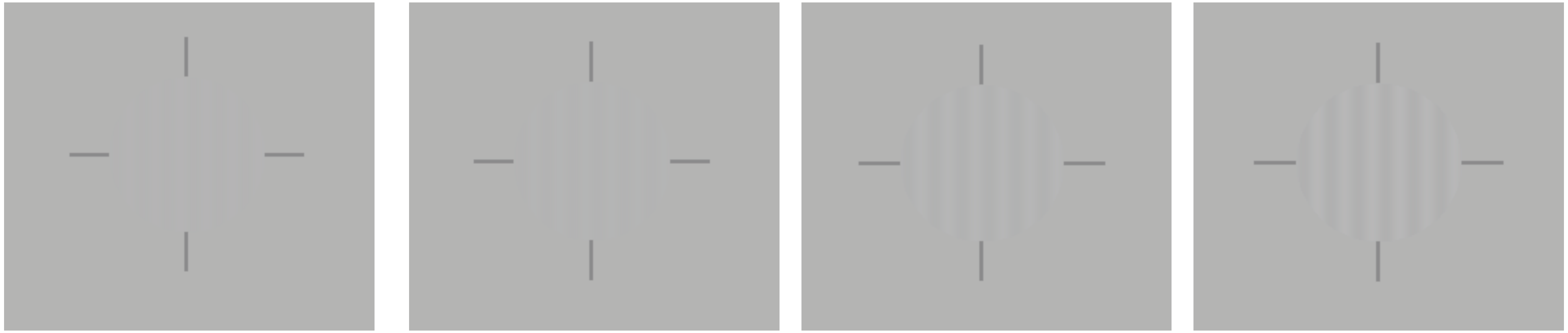
- Method of constant stimuli

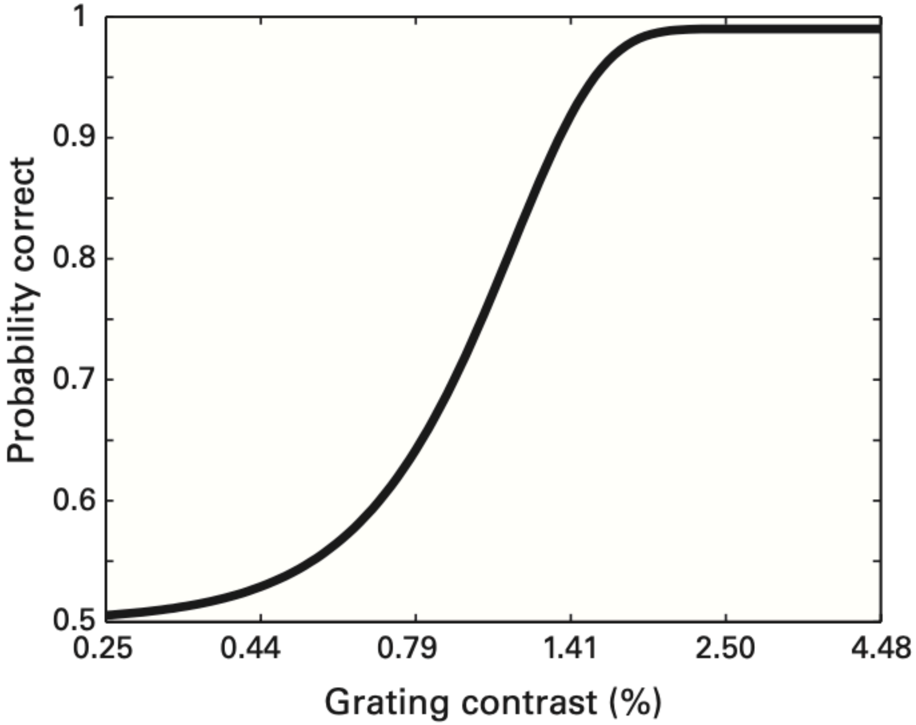




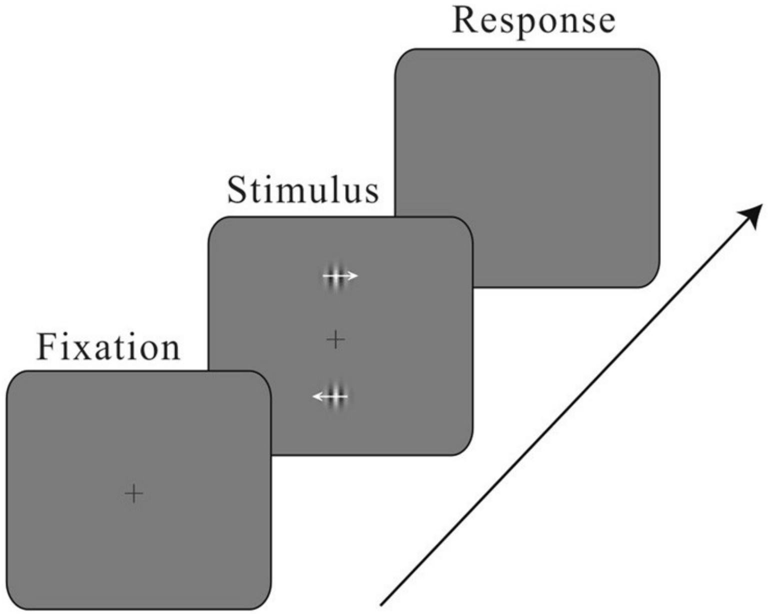
```
% Run nTrials trials
for i = 1 : nTrials
    con = p.contrasts(rec(i, 2));  % use contrast index from
                   % rec to set contrast for this trial
    Screen('DrawTexture', windowPtr, tex, [], [], 0, [], ...
           [], [], [], [], [180 sf con 0]);
        % draw the sine grating with phase 180, spatial
        % frequency, and contrast
    Screen('DrawLines', windowPtr, fixXY, 3, 0.3);
        % add the fixation crosshairs
    t0 = Screen('Flip', windowPtr, Secs + p.ISI);
        % show the stimulus and return the time
    Screen('Flip', windowPtr, t0 + p.stimDuration);
        % turn off the stimulus by flipping to background
      % image after p.stimDuration secs
      Screen('FillOval', windowPtr, 0, fixRect);
      % draw the smaller centered fixation
    Screen('Flip', windowPtr, t0 + 0.25 + p.stimDuration);
      % show small fixation briefly to cue the observer to
      % respond with the interval
```

Method of Constant Stimuli

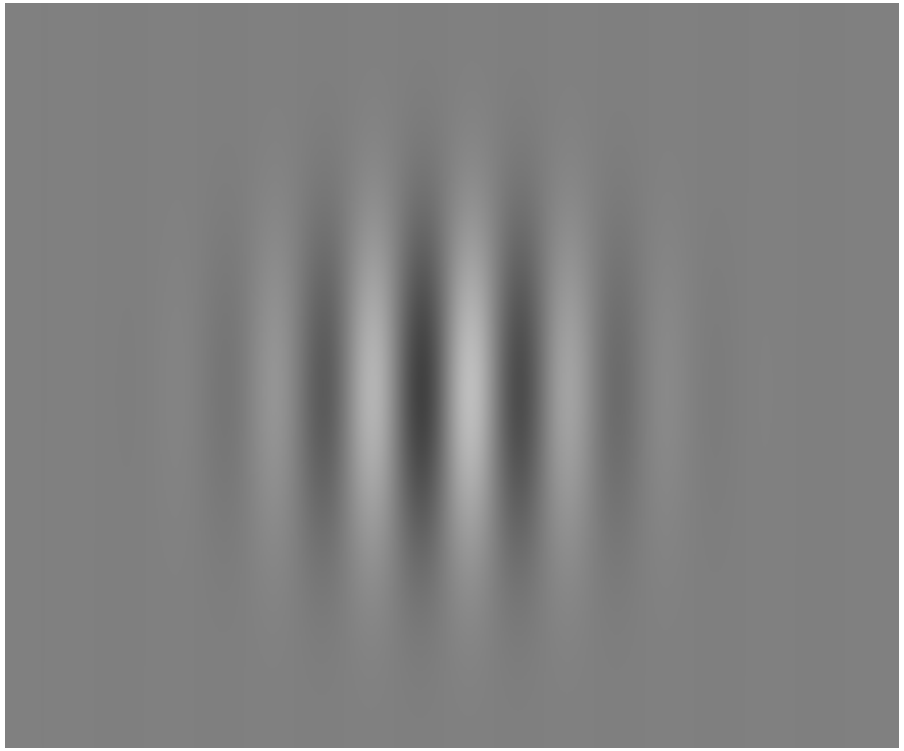




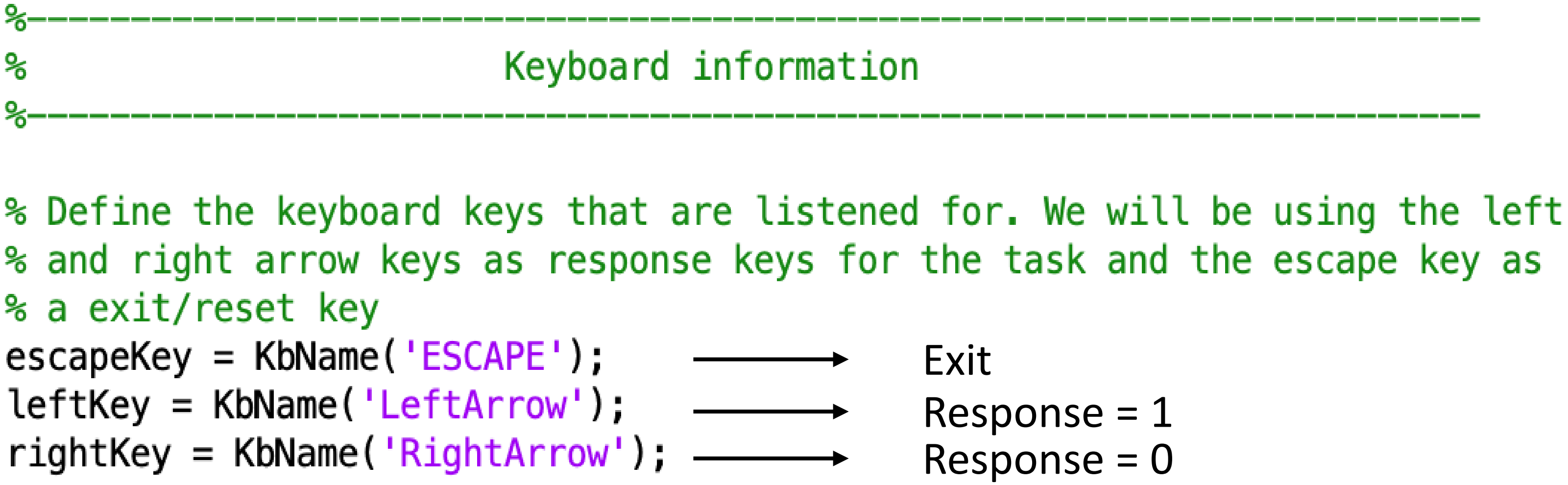
- Threshold Demo
- Attention task



```
%--------------------
% Gabor information
%--------------------

% Dimension of the region where will draw the Gabor in pixels
gaborDimPix = 300;

% Sigma of Gaussian
sigma = gaborDimPix / 7;

% Obvious Parameters
orientation = 90;
contrast = 0.5;
aspectRatio = 1.0;

% Spatial Frequency (Cycles Per Pixel)
% One Cycle = Grey-Black-Grey-White-Grey i.e. One Black and One White Lobe
numCycles = 8;
freq = numCycles / gaborDimPix;

% Build a procedural gabor texture
gabortex = CreateProceduralGabor(window, gaborDimPix, gaborDimPix, [],...
    [0.5 0.5 0.5 0.0], 1, 0.5);

% We will be displaying our Gabors either above or below fixation by 250
% pixels. We therefore have to determine these two locations in screen
% coordianates.
pixShift = 250;
xPos = [xCenter xCenter];
yPos = [yCenter - pixShift yCenter + pixShift];

% Count how many Gabors there are (two for this demo)
nGabors = numel(xPos);

% Make the destination rectangles for  the Gabors in the array i.e.
% rectangles the size of our Gabors cenetred above an below fixation.
baseRect = [0 0 gaborDimPix gaborDimPix];
allRects = nan(4, nGabors);
for i = 1:nGabors
    allRects(:, i) = CenterRectOnPointd(baseRect, xPos(i), yPos(i));
end

% Randomise the phase of the Gabors and make a properties matrix.
phaseLine = rand(1, nGabors) .* 360;
propertiesMat = repmat([NaN, freq, sigma, contrast,...
    aspectRatio, 0, 0, 0], nGabors, 1);
propertiesMat(:, 1) = phaseLine';
```



Keyboard Response



```
%-----------------------------------------------------------------------
%                       Keyboard information
%-----------------------------------------------------------------------

% Define the keyboard keys that are listened for. We will be using the left
% and right arrow keys as response keys for the task and the escape key as
% a exit/reset key
escapeKey = KbName('ESCAPE');
leftKey = KbName('LeftArrow');
rightKey = KbName('RightArrow');
```

Exit

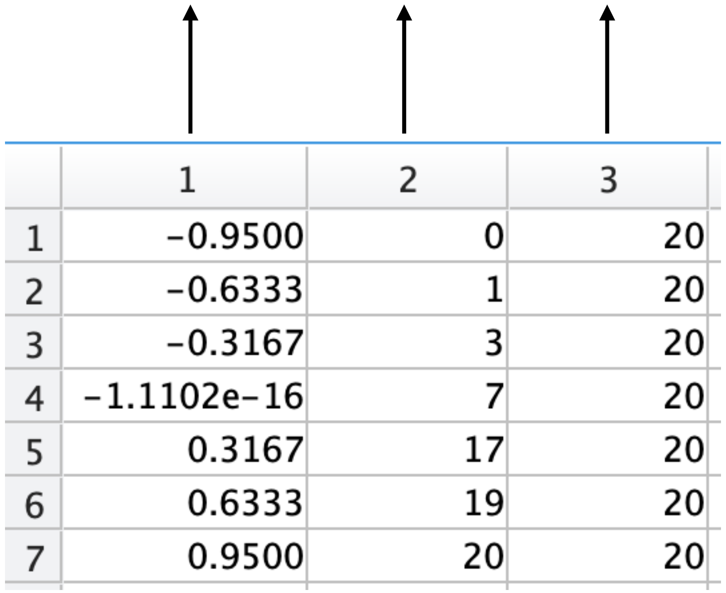
Response = 1

Response = 0

Collecting response

```
% Now we wait for a keyboard button signaling the observers response.
% The left arrow key signals a "left" response and the right arrow key
% a "right" response. You can also press escape if you want to exit the
% program
respToBeMade = true;
while respToBeMade
    [keyIsDown,secs, keyCode] = KbCheck;
    if keyCode(escapeKey)
        ShowCursor;
        sca;
        return
    elseif keyCode(leftKey)
        response = 1;
        respToBeMade = false;
    elseif keyCode(rightKey)
        response = 0;
        respToBeMade = false;
    end
end

% Record the response
respVector(stimValues == theAngle) = respVector(stimValues == theAngle)...
    + response;

% Add one to the counter for that stimulus
countVector(stimValues == theAngle) = countVector(stimValues == theAngle) + 1;
```

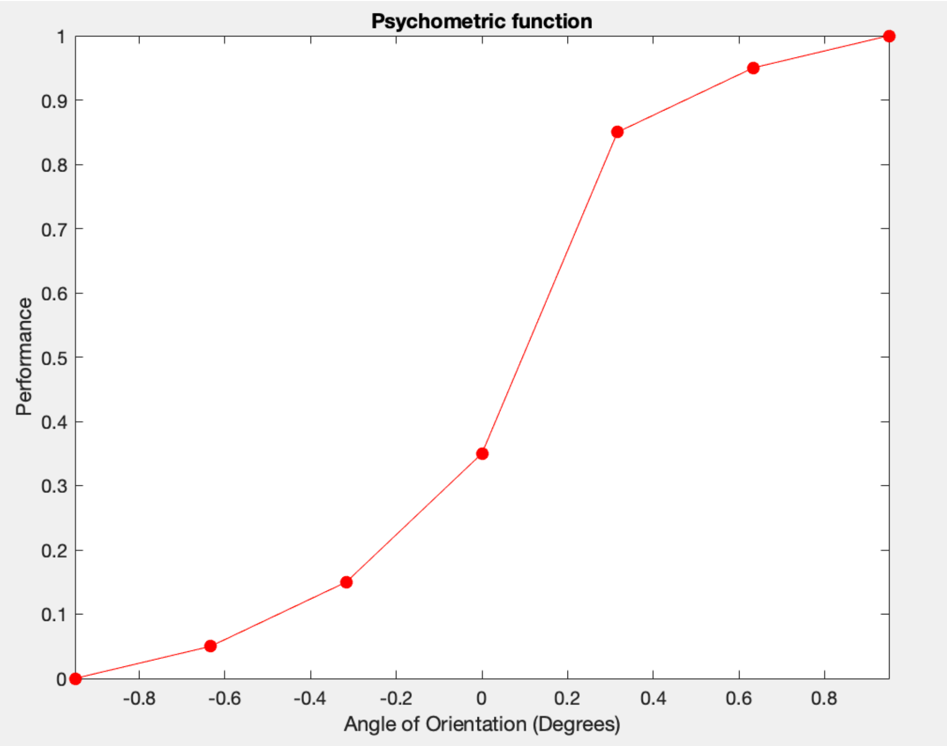
```
data = [stimValues; respVector; countVector]';
```



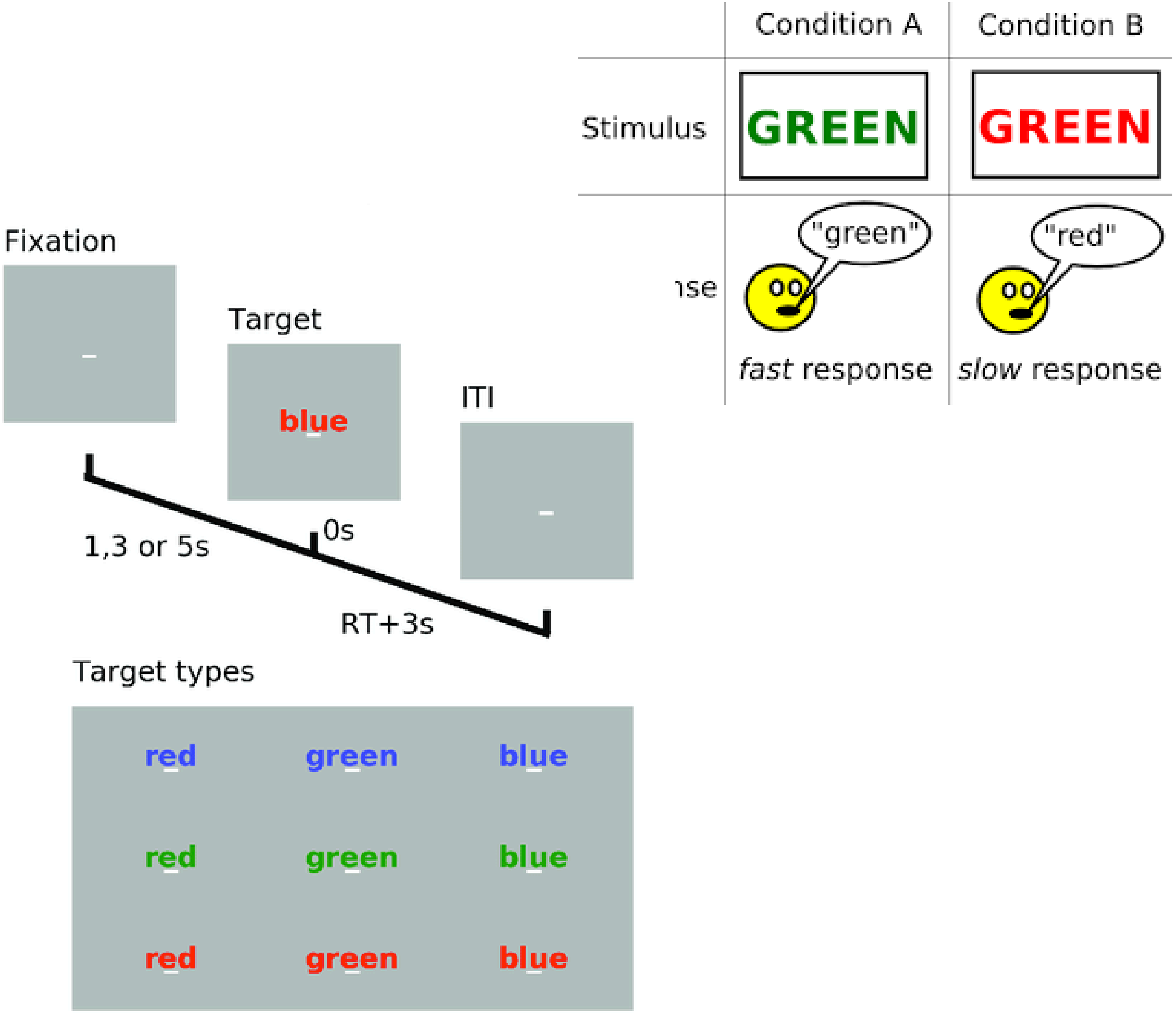
	1	2	3
1	-0.9500	0	20
2	-0.6333	1	20
3	-0.3167	3	20
4	-1.1102e-16	7	20
5	0.3167	17	20
6	0.6333	19	20
7	0.9500	20	20

Plot psychometric function

```
figure;
plot(data(:, 1), data(:, 2) ./ data(:, 3), 'ro-', 'MarkerFaceColor', 'r');
axis([min(data(:, 1)) max(data(:, 1)) 0 1]);
xlabel('Angle of Orientation (Degrees)');
ylabel('Performance');
title('Psychometric function');
```

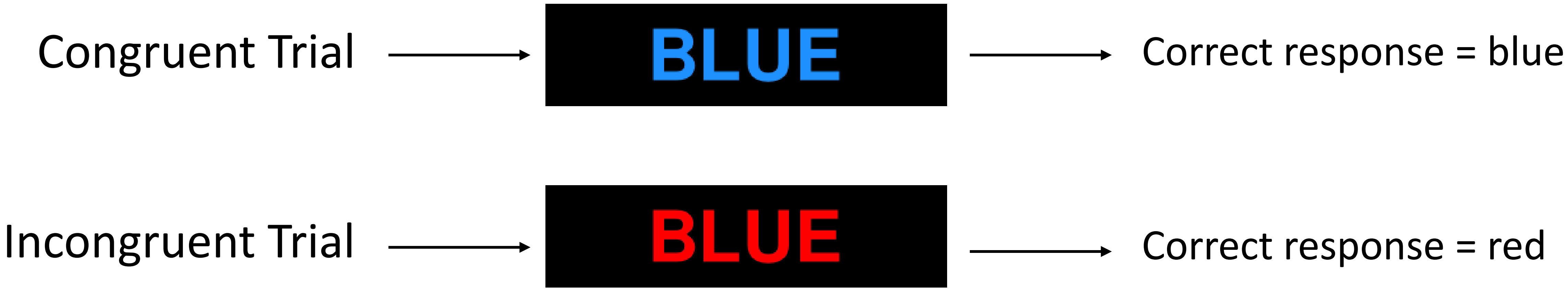


- Stroop Demo


Condition A
Condition B
Stimulus
GREEN
GREEN
ıse
"green"
"red"
fast response
slow response
Fixation
Target
blue
ITI
1,3 or 5s
0s
RT+3s
Target types
red green blue
red green blue
red green blue

Stroop Demo

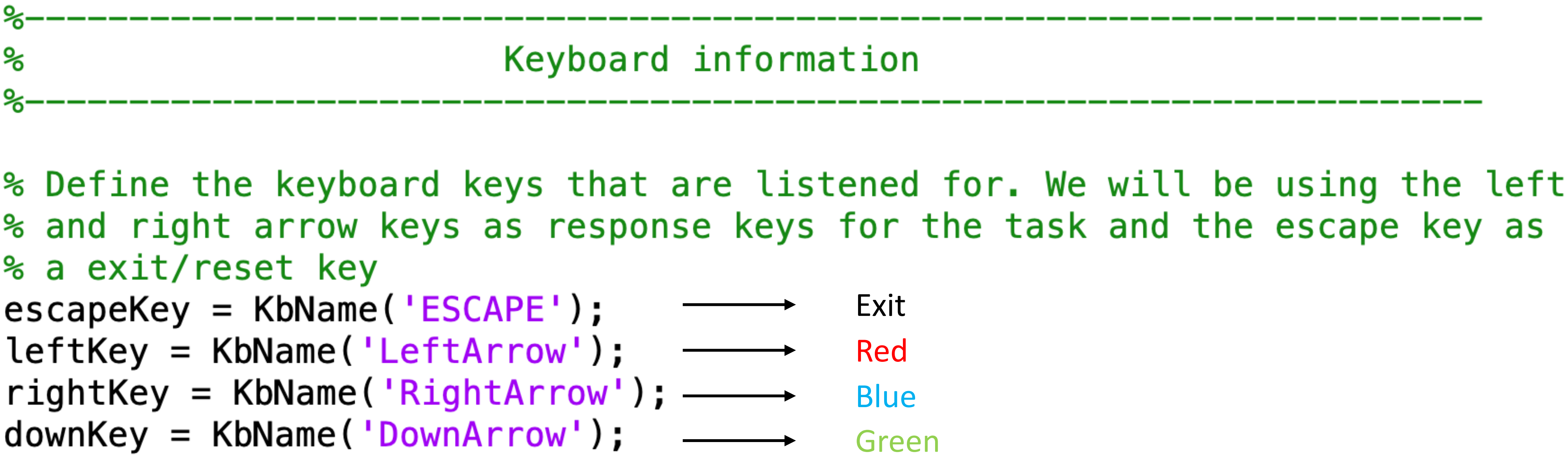
- The Stroop phenomenon demonstrates that it is difficult to name the ink color of a color word if there is a mismatch between ink color and word.



Congruent Trial → BLUE → Correct response = blue

Incongruent Trial → BLUE → Correct response = red

Keyboard Response



```
%----------------------------------------------------------------------
%                        Keyboard information
%----------------------------------------------------------------------

% Define the keyboard keys that are listened for. We will be using the left
% and right arrow keys as response keys for the task and the escape key as
% a exit/reset key
escapeKey = KbName('ESCAPE');
leftKey = KbName('LeftArrow');
rightKey = KbName('RightArrow');
downKey = KbName('DownArrow');
```

Exit

Red

Blue

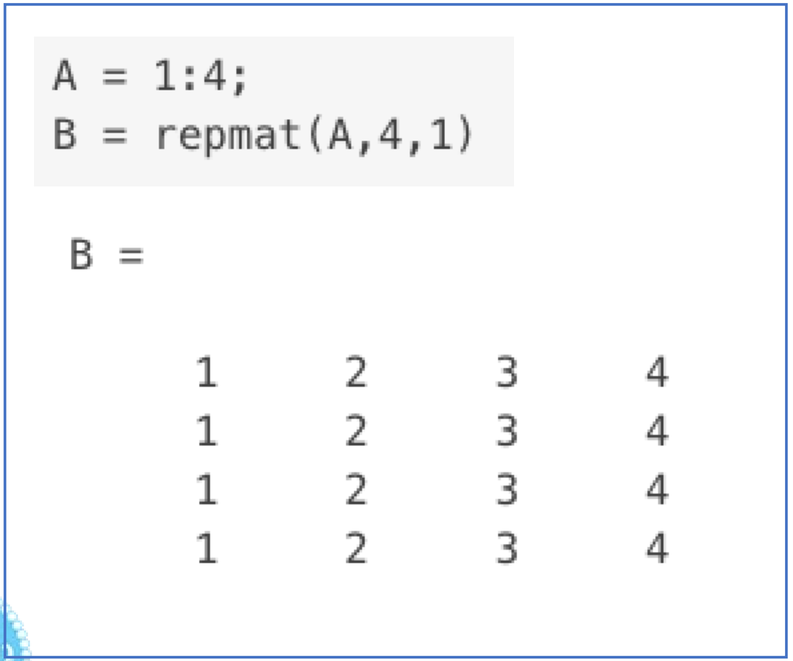
Green

Repmat:

B = repmat(A,m,n)

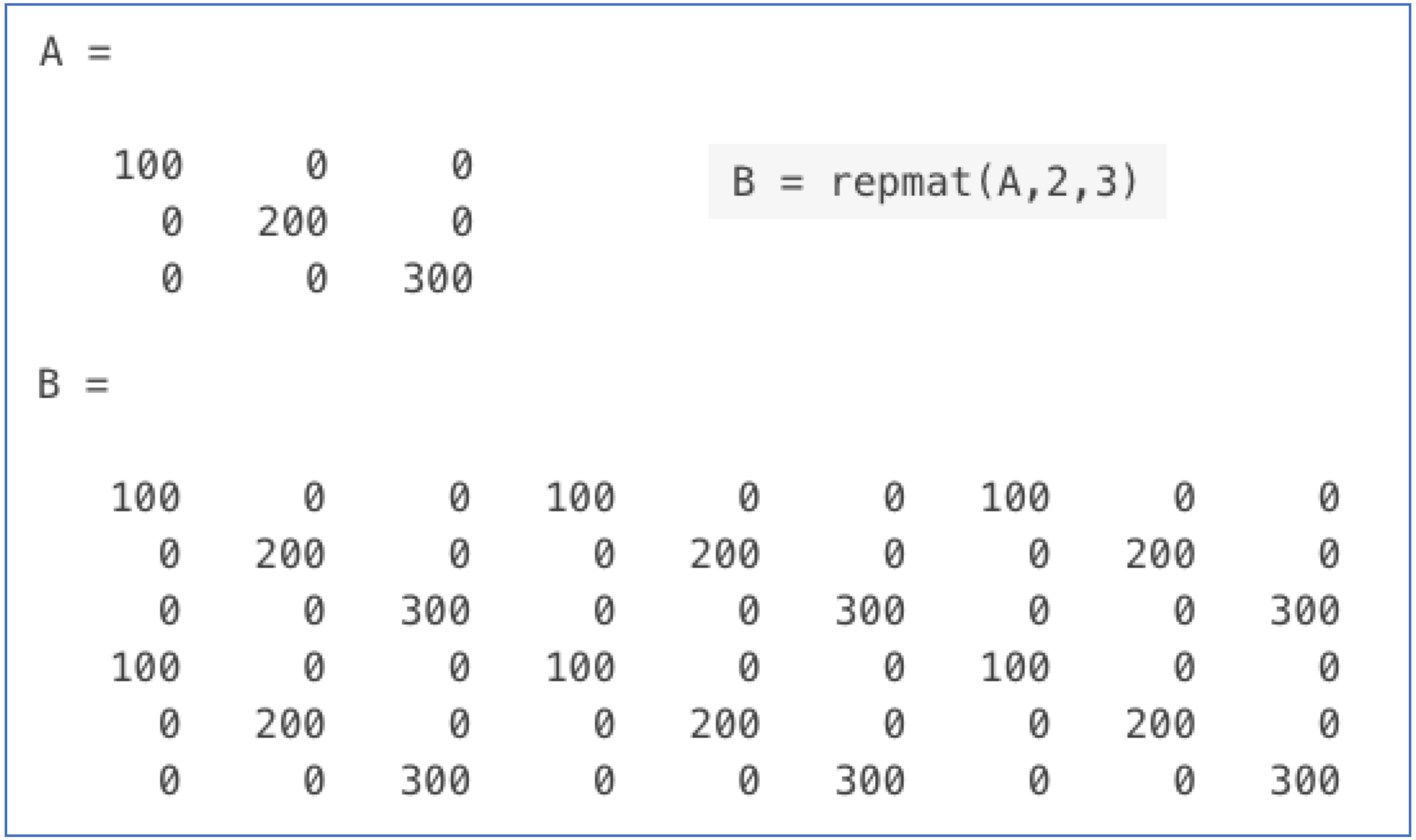
Repeat copies of A into a m-by-n block arrangement.

Example:



```
A = 1:4;
B = repmat(A,4,1)
```

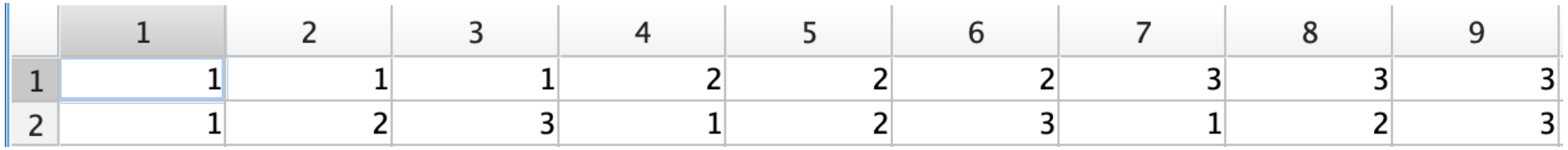
```
B =

     1     2     3     4
     1     2     3     4
     1     2     3     4
     1     2     3     4
```



```
B = repmat(A,2,3)
```

```
A =

   100     0     0
     0   200     0
     0     0   300

B =

   100     0     0   100     0     0   100     0     0
     0   200     0     0   200     0     0   200     0
     0     0   300     0     0   300     0     0   300
   100     0     0   100     0     0   100     0     0
     0   200     0     0   200     0     0   200     0
     0     0   300     0     0   300     0     0   300
```

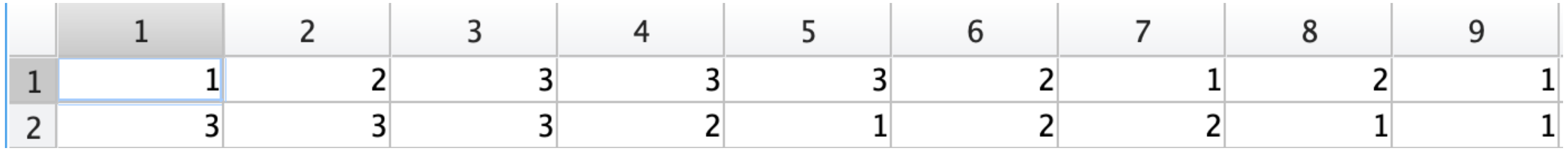
condMatrixBase



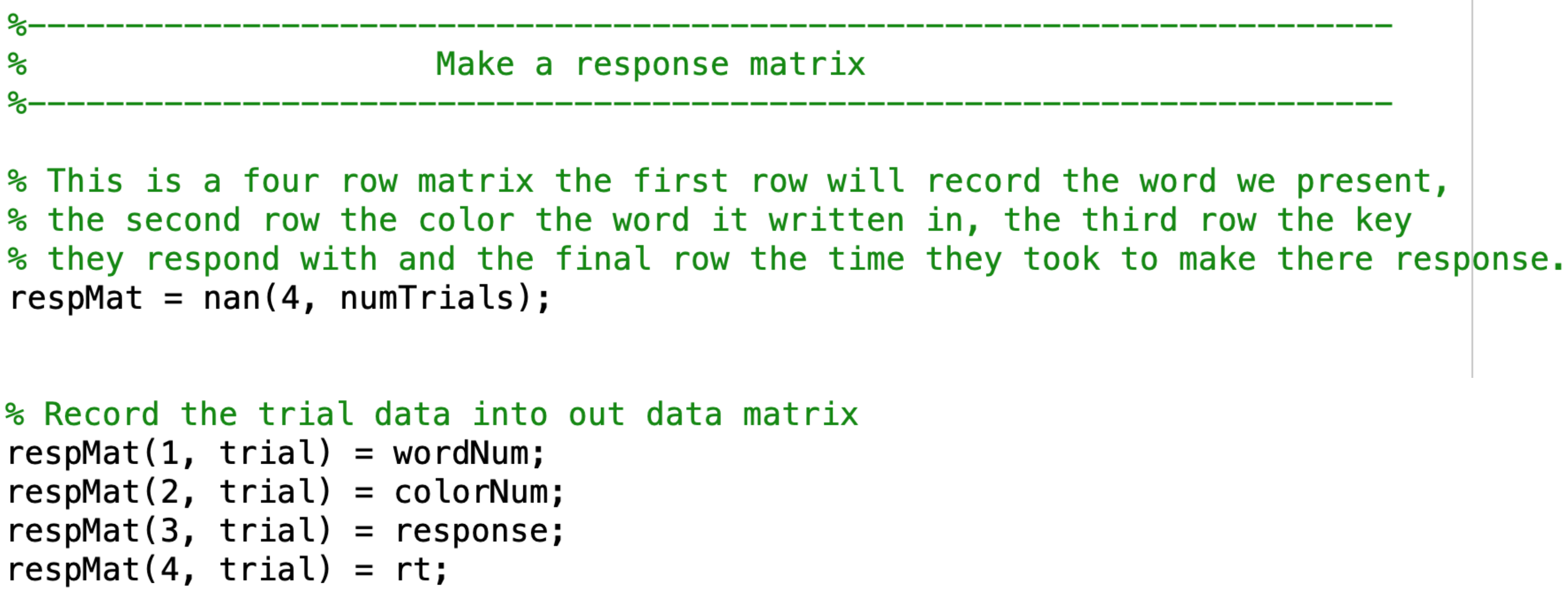
	1	2	3	4	5	6	7	8	9
1	1	1	1	2	2	2	3	3	3
2	1	2	3	1	2	3	1	2	3

```
%-----------------------------------------------------------------------
%                       Colors in words and RGB
%-----------------------------------------------------------------------

% We are going to use three colors for this demo. Red, Green and blue.
wordList = {'Red', 'Green', 'Blue'};
rgbColors = [1 0 0; 0 1 0; 0 0 1];

% Make the matrix which will determine our condition combinations
condMatrixBase = [sort(repmat([1 2 3], 1, 3)); repmat([1 2 3], 1, 3)];

% Number of trials per condition. We set this to one for this demo, to give
% us a total of 9 trials.
trialsPerCondition = 1;

% Duplicate the condition matrix to get the full number of trials
condMatrix = repmat(condMatrixBase, 1, trialsPerCondition);

% Get the size of the matrix
[~, numTrials] = size(condMatrix);

% Randomise the conditions
shuffler = Shuffle(1:numTrials);
condMatrixShuffled = condMatrix(:, shuffler);
```

condMatrixShuffled

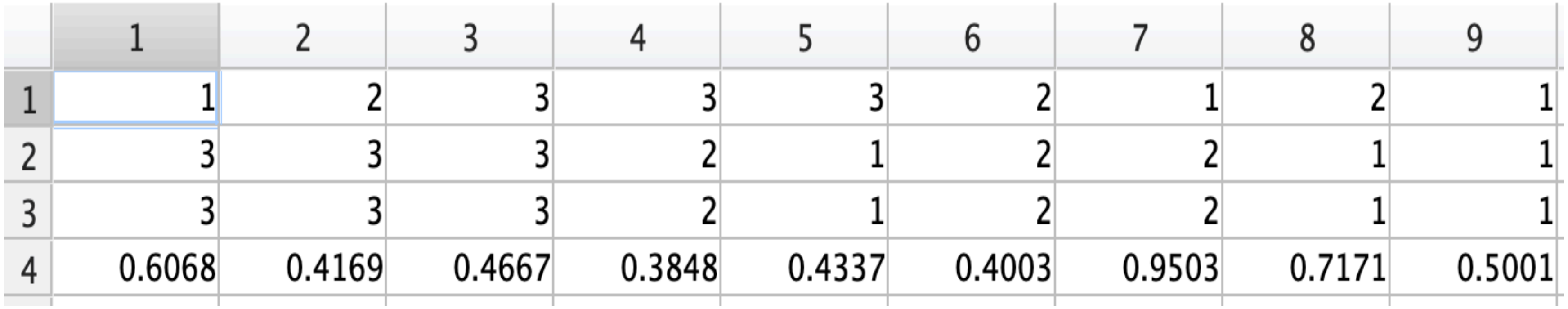


	1	2	3	4	5	6	7	8	9
1	1	2	3	3	3	2	1	2	1
2	3	3	3	2	1	2	2	1	1



```
%-------------------------------------------------------------------
%                     Make a response matrix
%-------------------------------------------------------------------

% This is a four row matrix the first row will record the word we present,
% the second row the color the word it written in, the third row the key
% they respond with and the final row the time they took to make there response.
respMat = nan(4, numTrials);


% Record the trial data into out data matrix
respMat(1, trial) = wordNum;
respMat(2, trial) = colorNum;
respMat(3, trial) = response;
respMat(4, trial) = rt;
```

respMat

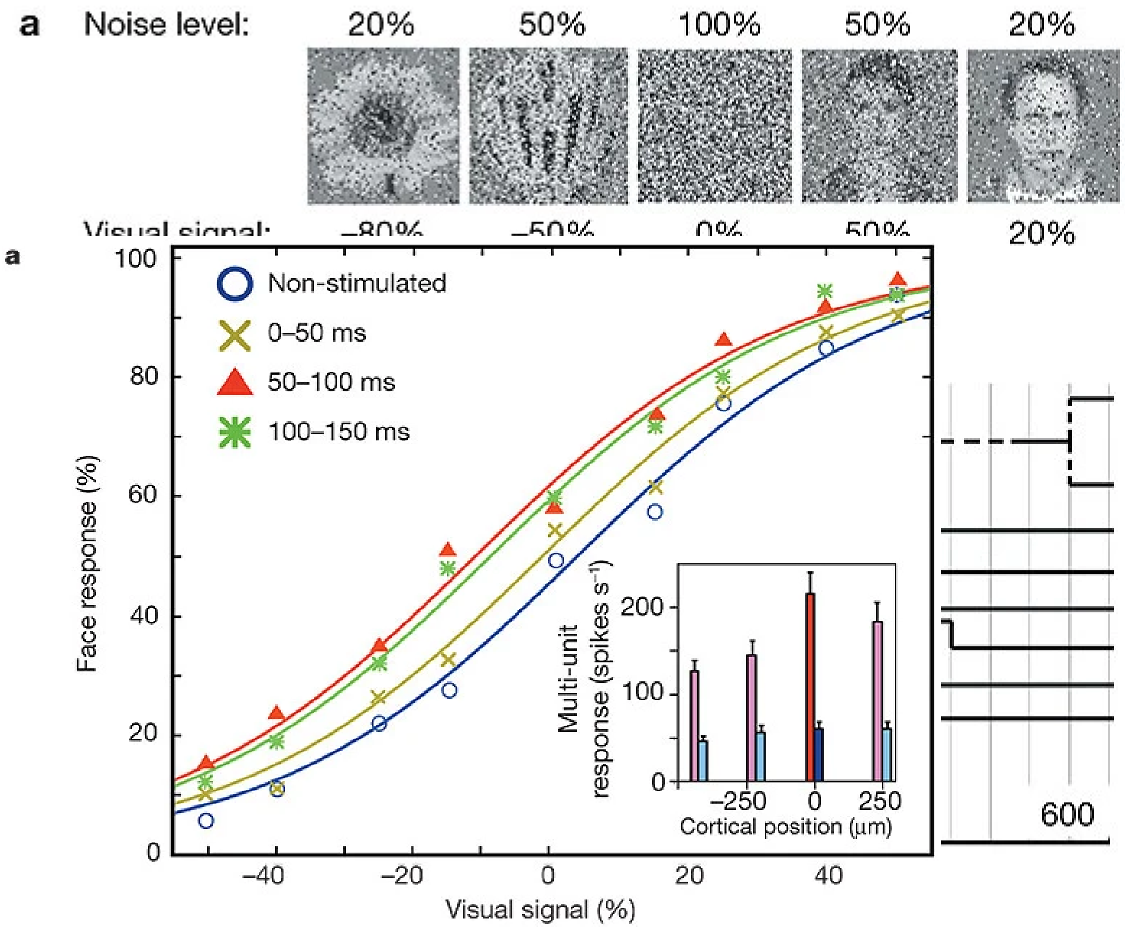


		1	2	3	4	5	6	7	8	9
Word number →	1	1	2	3	3	3	2	1	2	1
color number →	2	3	3	3	2	1	2	2	1	1
response →	3	3	3	3	2	1	2	2	1	1
Reaction time →	4	0.6068	0.4169	0.4667	0.3848	0.4337	0.4003	0.9503	0.7171	0.5001

Assignment#12

- Face detection task

- 500ms fixation
- 300 ms stimulus
- 500 ms delay
- Response collection
- Plot Psychometric function



Afraz, S. R., Kiani, R., & Esteky, H. (2006). Microstimulation of inferotemporal cortex influences face categorization. *Nature*, *442*(7103), 692-695.