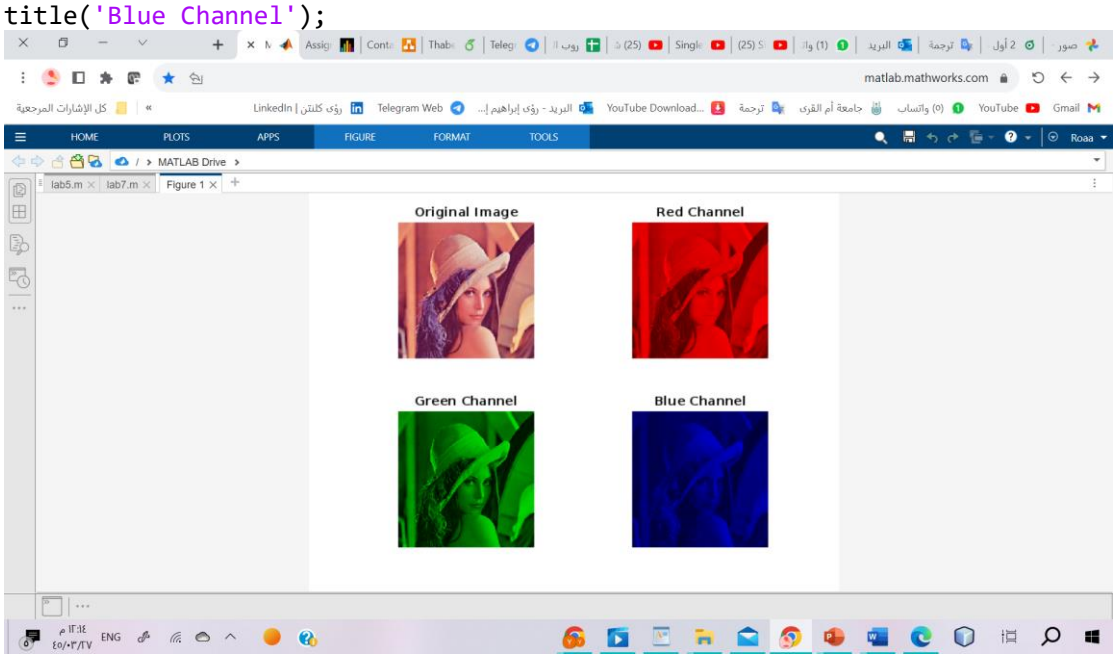
```
%lab 7
LRGB =imread("/MATLAB Drive/standard_test_images/lena_color_256.tif");
fR=zeros(size(LRGB,1),size(LRGB,2),size(LRGB,2),'uint8');
fG=zeros(size(LRGB,1),size(LRGB,2),size(LRGB,2),'uint8');
fB=zeros(size(LRGB,1),size(LRGB,2),size(LRGB,2),'uint8');

fR(:,:,1)=LRGB(:,:,1);
fG(:,:,2)=LRGB(:,:,2);
fB(:,:,3)=LRGB(:,:,3);

figure;
montage({LRGB,fR,fG,fB});
```

1. LRGB = imread("/MATLAB Drive/standard_test_images/lena_color_256.tif");
    - This line reads the image file "lena_color_256.tif" located in the "/MATLAB Drive/standard_test_images/" directory and assigns it to the variable LRGB.
2. fR = zeros(size(LRGB,1),size(LRGB,2),size(LRGB,2),'uint8');
    - This line creates an empty matrix fR with the same number of rows and columns as LRGB and with a depth of 3 (representing the RGB color channels). The data type is set to 'uint8', which represents unsigned 8-bit integers.
3. fG = zeros(size(LRGB,1),size(LRGB,2),size(LRGB,2),'uint8');
    - This line creates an empty matrix fG with the same size and data type as fR.
4. fB = zeros(size(LRGB,1),size(LRGB,2),size(LRGB,2),'uint8');
    - This line creates an empty matrix fB with the same size and data type as fR.
5. fR(:,:,1) = LRGB(:,:,1);
    - This line assigns the red color channel of LRGB to the corresponding channel of fR. It copies the red channel data from LRGB into the first channel of fR.
6. fG(:,:,2) = LRGB(:,:,2);
    - This line assigns the green color channel of LRGB to the corresponding channel of fG. It copies the green channel data from LRGB into the second channel of fG.
7. fB(:,:,3) = LRGB(:,:,3);
    - This line assigns the blue color channel of LRGB to the corresponding channel of fB. It copies the blue channel data from LRGB into the third channel of fB.
8. figure;
    - This line creates a new figure window for displaying the montage.
9. montage({LRGB, fR, fG, fB});

- This line creates a montage of the images specified in the cell array. It displays the original image LRGB along with the separate color channels fR, fG, and fB as subplots in the figure window.

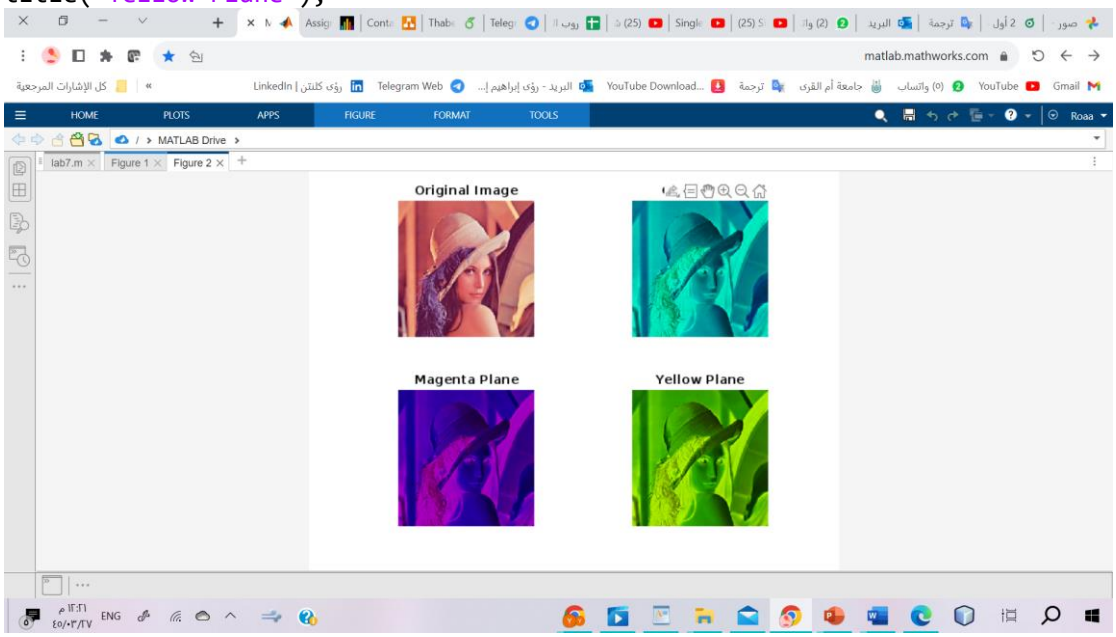
The code essentially separates the color channels of the input image LRGB into individual matrices fR, fG, and fB. Then it creates a montage of these images to visualize the separate color channels alongside the original image.

```
LRGB = imread("/MATLAB Drive/standard_test_images/lena_color_256.tif");
fR = zeros(size(LRGB, 1), size(LRGB, 2), size(LRGB, 3), 'uint8');
fG = zeros(size(LRGB, 1), size(LRGB, 2), size(LRGB, 3), 'uint8');
fB = zeros(size(LRGB, 1), size(LRGB, 2), size(LRGB, 3), 'uint8');

fR(:, :, 1) = LRGB(:, :, 1);
fG(:, :, 2) = LRGB(:, :, 2);
fB(:, :, 3) = LRGB(:, :, 3);

figure;
subplot(2, 2, 1);
imshow(LRGB);
title('Original Image');

subplot(2, 2, 2);
imshow(fR);
title('Red Channel');

subplot(2, 2, 3);
imshow(fG);
title('Green Channel');

subplot(2, 2, 4);
imshow(fB);
title('Blue Channel');
```



The code reads an image file called "lena_color_256.tif" and assigns it to the variable LRGB. It then creates three empty matrices fR, fG, and fB with the same size as LRGB, and of data type 'uint8'. The red color channel of LRGB is copied to fR, the green channel to fG, and the blue channel to fB. Finally, it displays a montage of the original image LRGB and the separate color channels fR, fG, and fB in a figure window.

```
%%%%Colors CMY planes%%%%%
% Create custom color matrices for CMY planes
C = cat(3, zeros(size(LRGB, 1), size(LRGB, 2)), 255 - LRGB(:, :, 2), 255 -
LRGB(:, :, 3));
M = cat(3, 255 - LRGB(:, :, 1), zeros(size(LRGB, 1), size(LRGB, 2)), 255 -
LRGB(:, :, 3));
Y = cat(3, 255 - LRGB(:, :, 1), 255 - LRGB(:, :, 2), zeros(size(LRGB, 1),
size(LRGB, 2)));

% Display the CMY color planes
figure;
subplot(2, 2, 1);
imshow(LRGB);
title('Original Image');

subplot(2, 2, 2);
imshow(C);
title('Cyan Plane');

subplot(2, 2, 3);
imshow(M);
title('Magenta Plane');

subplot(2, 2, 4);
imshow(Y);
title('Yellow Plane');
```



C = cat(3, zeros(size(LRGB, 1), size(LRGB, 2)), 255 - LRGB(:, :, 2), 255 - LRGB(:, :, 3));

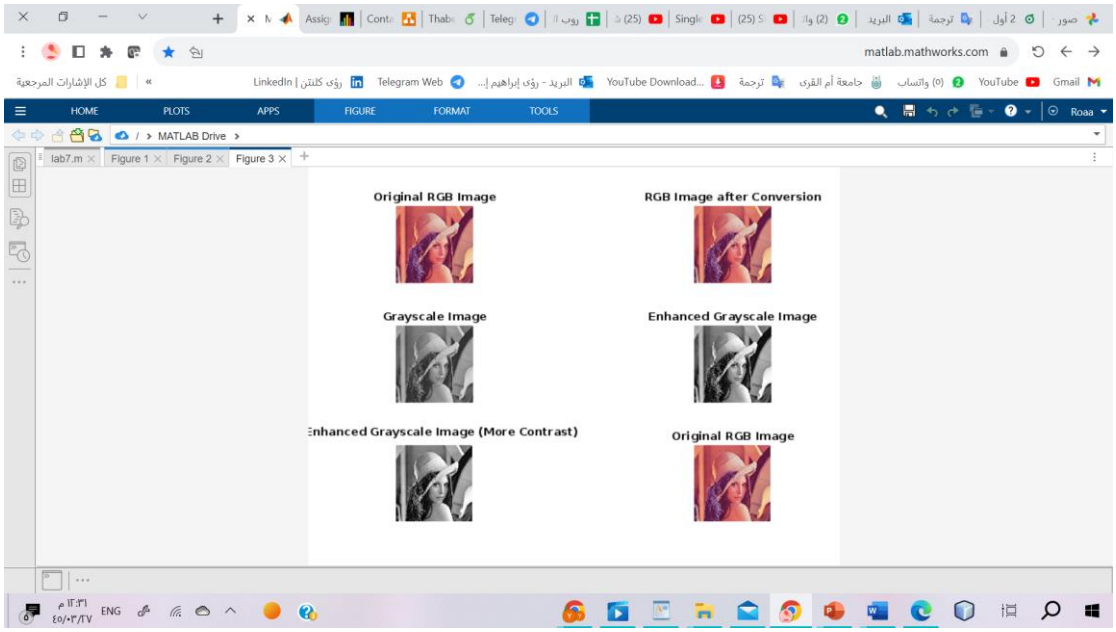
- Create a custom color matrix for the Cyan plane by concatenating three matrices: a matrix of zeros, the complement of the green channel, and the complement of the blue channel.

M = cat(3, 255 - LRGB(:, :, 1), zeros(size(LRGB, 1), size(LRGB, 2)), 255 - LRGB(:, :, 3));

- Create a custom color matrix for the Magenta plane by concatenating three matrices: the complement of the red channel, a matrix of zeros, and the complement of the blue channel.

Y = cat(3, 255 - LRGB(:, :, 1), 255 - LRGB(:, :, 2), zeros(size(LRGB, 1), size(LRGB, 2)));

- Create a custom color matrix for the Yellow plane by concatenating three matrices: the complement of the red channel, the complement of the green channel, and a matrix of zeros.




```
%Conversion between RGB and YIQ
% Use rgb2ntsc () to convert RGB to YQI
% ntsc2rgb() to convert YIQ to RGB

% Convert RGB to YIQ
YIQ = rgb2ntsc(LRGB);

% Convert YIQ back to RGB
RGB = ntsc2rgb(YIQ);

% Convert RGB to grayscale
gray = rgb2gray(LRGB);

% Enhance grayscale image
enhanced_gray = imadjust(gray);
```

```
% Display the images
figure;

subplot(3, 3, 1);
imshow(LRGB);
title('Original RGB Image');

subplot(3, 3, 3);
imshow(RGB);
title('RGB Image after Conversion');

subplot(3, 3, 4);
imshow(gray);
title('Grayscale Image');

subplot(3, 3, 6);
imshow(enhanced_gray);
title('Enhanced Grayscale Image');

subplot(3, 3, 7);
imshow(enhanced_gray);
title('_    Enhanced Grayscale Image (More Contrast)');

subplot(3, 3, 9);
imshow(LRGB);
title('Original RGB Image');
```

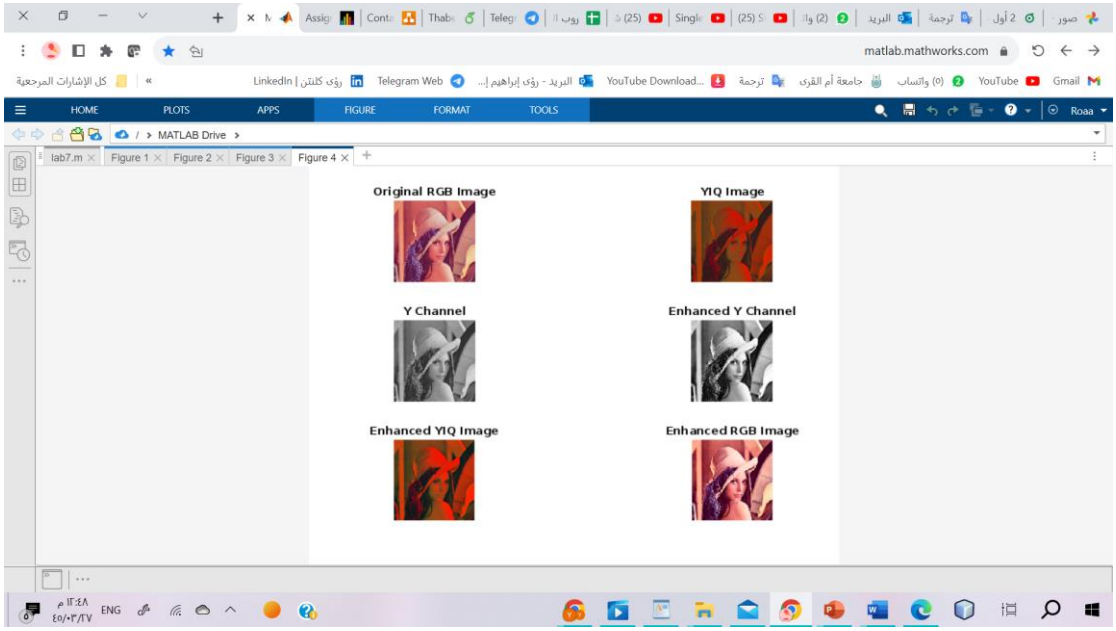
Explanation of the code:

1. The code reads an RGB image and assigns it to the variable LRGB.
2. The rgb2ntsc() function is used to convert the RGB image to YIQ color space, and the result is stored in the variable YIQ.
3. The ntsc2rgb() function is used to convert the YIQ image back to RGB, and the result is stored in the variable RGB.
4. The rgb2gray() function is used to convert the original RGB image to grayscale, and the result is stored in the variable gray.
5. The imadjust() function is applied to the grayscale image to enhance its contrast, and the result is stored in the variable enhanced_gray.
6. The images are displayed in a 2x3 grid of subplots using the imshow() function. The titles of each subplot indicate the type of image displayed.

The displayed images are as follows:

- Subplot 1: Original RGB Image
- Subplot 2: RGB Image after Conversion from YIQ
- Subplot 3: Grayscale Image
- Subplot 4: Enhanced Grayscale Image
- Subplot 5: Enhanced Grayscale Image with More Contrast

- Subplot 6: Original RGB Image (Repeating for comparison)




```
% Convert RGB to YIQ
fYIQ = rgb2ntsc(LRGB);
Y = fYIQ(:,:,1);

% Enhance Y channel using histogram equalization
New_Y = histeq(Y);

% Create a new YIQ image with the enhanced Y channel
New_YIQ = fYIQ;
New_YIQ(:,:,1) = New_Y;

% Convert the enhanced YIQ image back to RGB
New_RGB = ntsc2rgb(New_YIQ);

% Display the images
figure;

subplot(3, 3, 1);
imshow(LRGB);
title('Original RGB Image');

subplot(3, 3, 3);
imshow(fYIQ);
title('YIQ Image');

subplot(3, 3, 4);
imshow(Y);
title('Y Channel');

subplot(3, 3, 6);
imshow(New_Y);
title('Enhanced Y Channel');

subplot(3, 3, 7);
imshow(New_YIQ);
title('Enhanced YIQ Image');
```

```
subplot(3, 3, 9);
imshow(New_RGB);
title('Enhanced RGB Image');
```

Explanation of the code:

1. The code reads the RGB image and assigns it to the variable LRGB.
2. The rgb2ntsc() function is used to convert the RGB image to YIQ color space, and the result is stored in the variable fYIQ.
3. The code extracts the Y channel from the YIQ image and assigns it to the variable Y.
4. The histeq() function is applied to the Y channel to enhance its contrast, and the result is stored in the variable New_Y.
5. A new YIQ image is created (New_YIQ) by copying the original YIQ image and replacing the Y channel with the enhanced Y channel.
6. The ntsc2rgb() function is used to convert the enhanced YIQ image back to RGB, and the result is stored in the variable New_RGB.
7. The images are displayed in a 2x3 grid of subplots using the imshow() function. The titles of each subplot indicate the name of the corresponding image.

The displayed images are as follows:

- Subplot 1: Original RGB Image
- Subplot 2: YIQ Image
- Subplot 3: Y Channel of YIQ Image
- Subplot 4: Enhanced Y Channel
- Subplot 5: Enhanced YIQ Image
- Subplot 6: Enhanced RGB Image

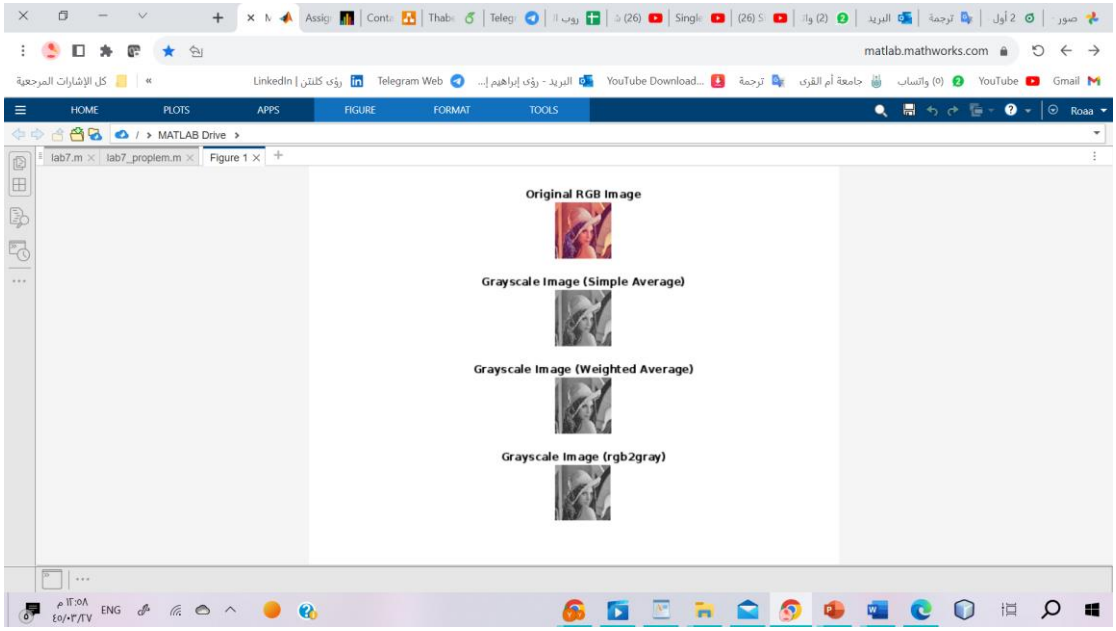
The purpose of this code is to demonstrate the enhancement of the Y channel in the YIQ color space. The Y channel represents the luminance component of the image, and by enhancing it, we can improve the overall image quality. The resulting enhanced Y channel is then used to construct a new YIQ image, which is finally converted back to RGB for display.

رؤى إبراهيم كلنتن شعبة 1 نظري
441004834 لاب 7

Problems:

1- Read lena color image (or any other color image) then show the grayvalue images computed from the images by averaging the corresponding red, green and blue pixels and the gray image computed by using the MATLAB command **rgb2gray**.
   a. Using simple average
   b. Weighted average

Show your results and write your comments about the differences.




```
LRGB = imread("/MATLAB Drive/standard_test_images/lena_color_256.tif");

% Compute grayscale image using simple average
gray_simple = uint8(mean(LRGB, 3));

% Compute grayscale image using weighted average
weights = [0.2989, 0.5870, 0.1140];
weights = reshape(weights, 1, 1, 3); % Reshape weights to match LRGB
dimensions
gray_weighted = uint8(sum(double(LRGB) .* weights, 3));

% Compute grayscale image using rgb2gray
gray_rgb2gray = rgb2gray(LRGB);

% Display the images
figure;

subplot(4, 3, 2);
imshow(LRGB);
title('Original RGB Image');

subplot(4, 3, 5);
imshow(gray_simple);
title('Grayscale Image (Simple Average)');

subplot(4, 3, 8);
imshow(gray_weighted);
title('Grayscale Image (Weighted Average)');

subplot(4, 3, 11);
imshow(gray_rgb2gray);
```

```
title('Grayscale Image (rgb2gray)');
```

1. The code reads an RGB image named "lena_color_256.tif" and assigns it to the variable LRGB.
2. The grayscale image using the simple average method is computed by taking the mean of the red, green, and blue channels of LRGB. The resulting grayscale image is stored in the variable gray_simple.
3. The weights for the weighted average method are defined as [0.2989, 0.5870, 0.1140]. These weights determine the contribution of each color channel to the grayscale image.
4. The weights array is reshaped to match the dimensions of the LRGB array. This ensures compatibility for element-wise multiplication later on.
5. The grayscale image using the weighted average method is computed by multiplying each channel of LRGB with the corresponding weights, summing them up, and converting the result to uint8 format. The resulting grayscale image is stored in the variable gray_weighted.
6. The grayscale image using the rgb2gray command is computed by applying the rgb2gray() function to the LRGB image. This function converts the RGB image to grayscale using a specific formula.
7. The images are displayed in a figure with four rows and three columns of subplots.

- The original RGB image is displayed in the second subplot of the first row.
- The grayscale image using the simple average method is displayed in the second subplot of the second row.
- The grayscale image using the weighted average method is displayed in the second subplot of the third row.
- The grayscale image using the rgb2gray command is displayed in the second subplot of the fourth row.

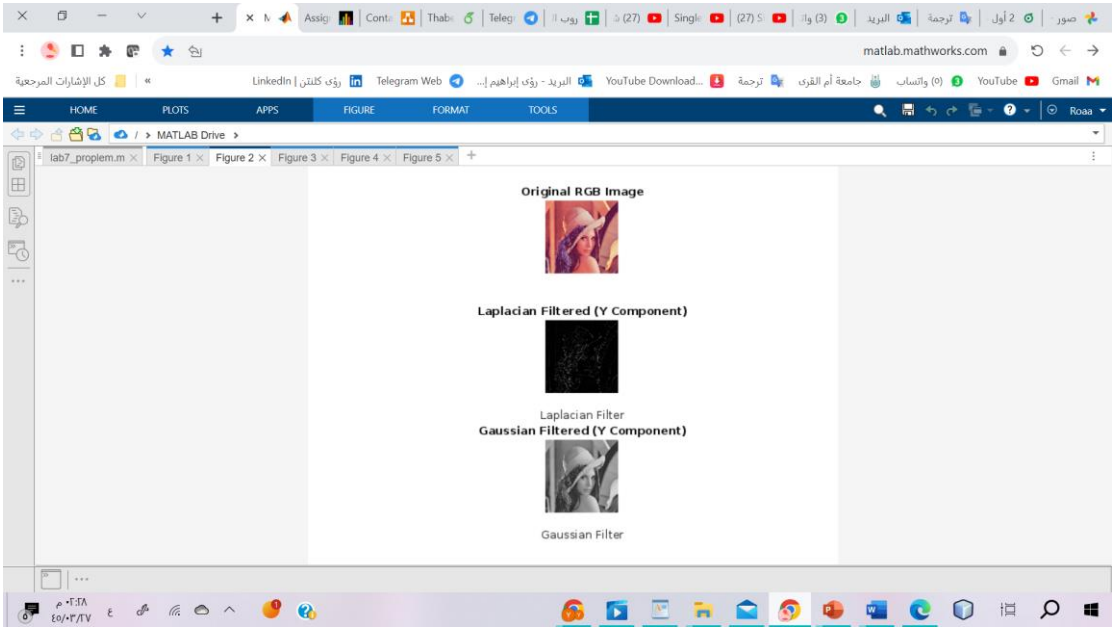
Each subplot is displayed using the imshow() function, and the title() function is used to set titles for each subplot.

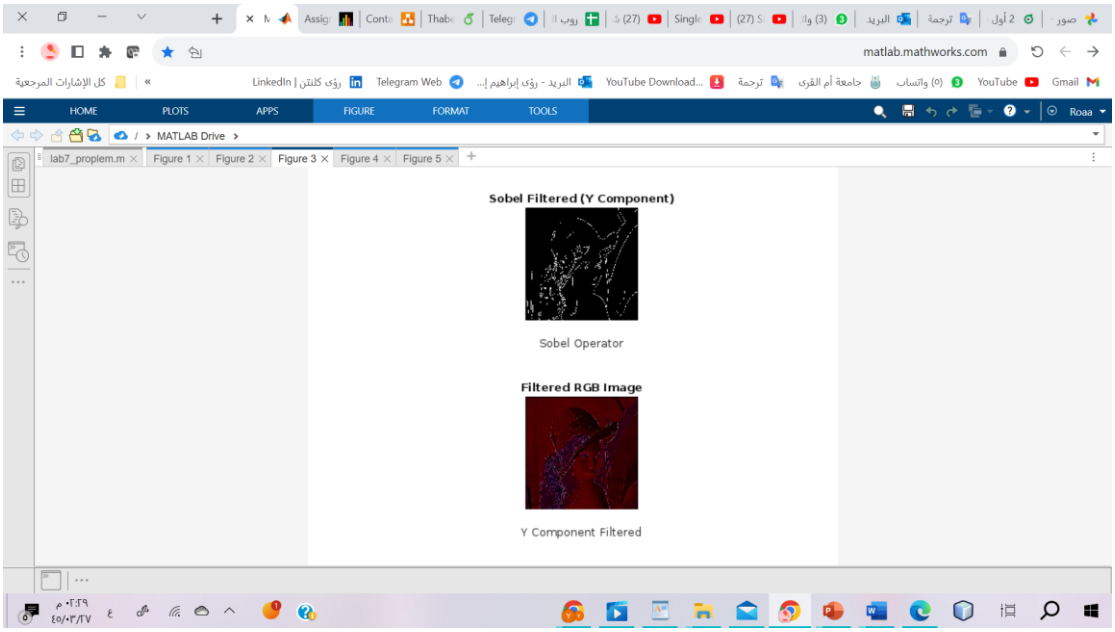
The purpose of this code is to compute and display different grayscale versions of the original RGB image for comparison.

رؤى إبراهيم كلنتن شعبة 1 نظري
441004834 لاب 7

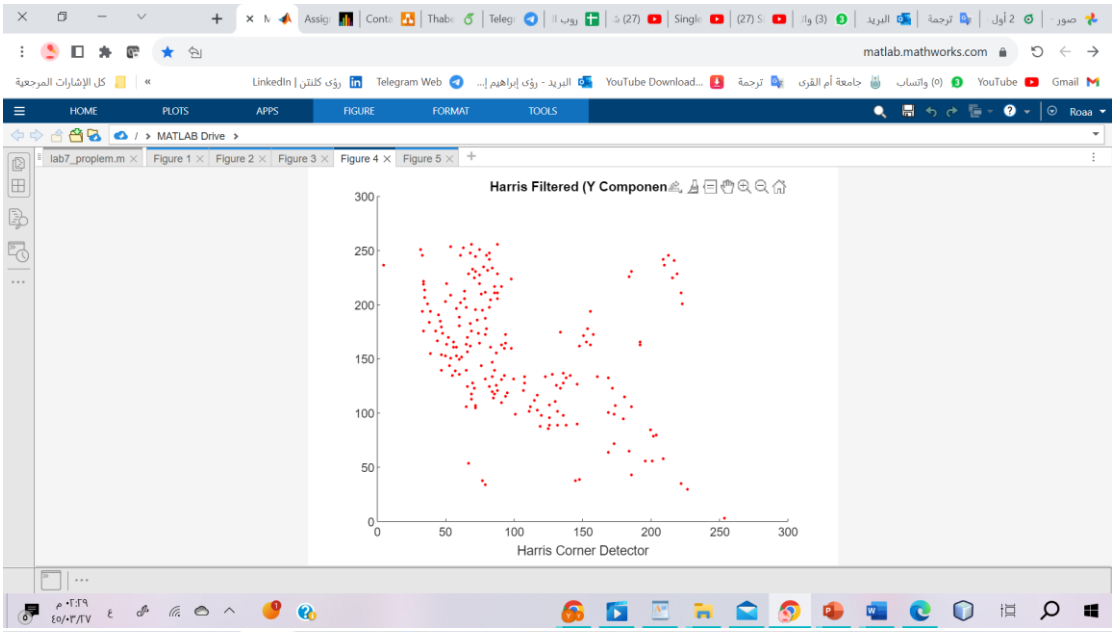
2- Convert lena image to YIQ then apply the following on the Y component only:
   a. Laplacian filter
   b. Gaussian filter
   c. Sobel operator
   d. Harris corner detector

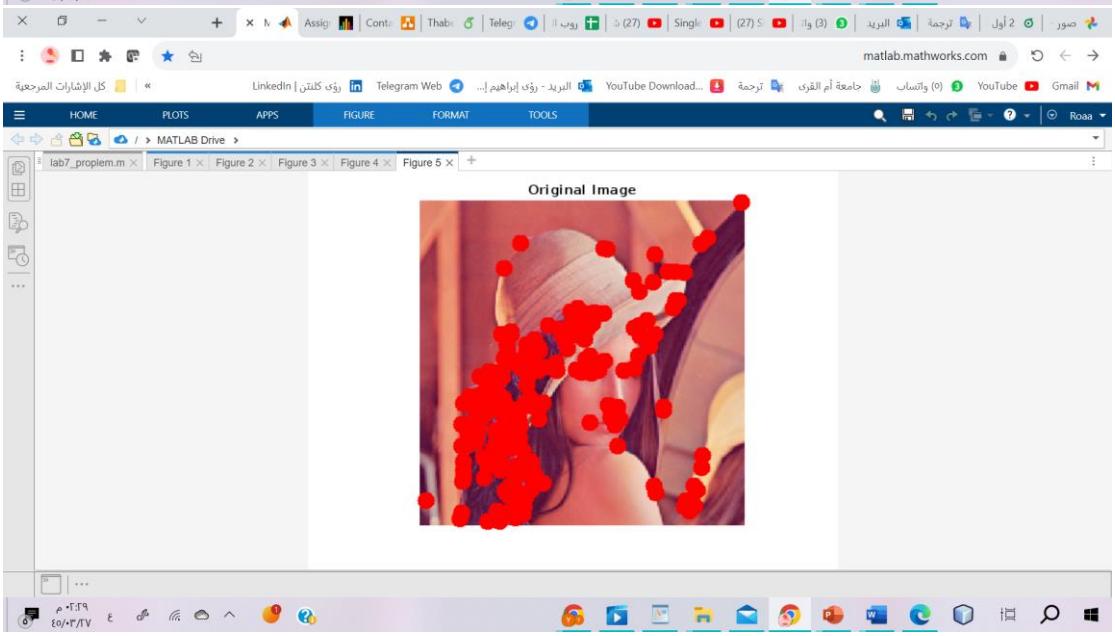
Then convent it back to RGB and show the results for each case





رؤى إبراهيم كلنتن شعبة 1 نظري
441004834 لاب 7






```
%q2
% Read the Lena image
Lena = imread("/MATLAB Drive/standard_test_images/lena_color_256.tif");

% Convert Lena image to YIQ color space
YIQ = rgb2ntsc(Lena);

% Extract the Y component
Y = YIQ(:,:,1);

% Part (a): Apply Laplacian filter to the Y component
laplacian_filtered = imfilter(Y, fspecial('laplacian'));

% Part (b): Apply Gaussian filter to the Y component
gaussian_filtered = imgaussfilt(Y);

% Part (c): Apply Sobel operator to the Y component
sobel_filtered = edge(Y, 'Sobel');
```

```
% Part (d): Apply Harris corner detector to the Y component
harris_filtered = corner(Y);

% Convert the filtered Y component back to RGB
YIQ_filtered = cat(3, laplacian_filtered, YIQ(:,:,2), YIQ(:,:,3));
RGB_filtered = ntsc2rgb(YIQ_filtered);

% Display the output images
figure;

subplot(3, 3, 2);
imshow(Lena);
title('Original RGB Image');

subplot(3, 3, 5);
imshow(laplacian_filtered);
title('Laplacian Filtered (Y Component)');
xlabel('Laplacian Filter');

subplot(3, 3, 8);
imshow(gaussian_filtered);
title('Gaussian Filtered (Y Component)');
xlabel('Gaussian Filter');

figure;
subplot(2, 3, 2);
imshow(sobel_filtered);
title('Sobel Filtered (Y Component)');
xlabel('Sobel Operator');

subplot(2, 3, 5);
imshow(RGB_filtered);
title('Filtered RGB Image');
xlabel('Y Component Filtered');

% Create a new figure for the scatter plot
figure;
scatter(harris_filtered(:,1), harris_filtered(:,2), 'r.', 'SizeData', 50);
% Increase the marker size to 50
title('Harris Filtered (Y Component)');
xlabel('Harris Corner Detector');

% Explanation:
% - Part (a): The Laplacian filter enhances edges and fine details in the Y
component of the image.
% - Part (b): The Gaussian filter smooths the Y component of the image,
reducing noise and blurring details.
% - Part (c): The Sobel operator detects edges in the Y component of the
image.
% - Part (d): The Harris corner detector identifies corners and points of
interest in the Y component of the image.
% - The filtered Y component is combined with the original U and V
components to obtain the filtered RGB image.

% Convert Lena image to grayscale
grayLena = rgb2gray(Lena);

% Apply Harris corner detector to the grayscale image
```

```
corners = detectHarrisFeatures(grayLena);

% Display the original image with corner points
figure;
imshow(Lena);
title('Original Image');
hold on;
plot(corners.Location(:,1), corners.Location(:,2), 'r.', 'MarkerSize', 50);
hold off;

% Explanation:
% - The grayscale version of the Lena image is used for the Harris corner
detection.
% - The detectHarrisFeatures function identifies corner points in the
image.
% - The original image is then displayed with the corner points overlaid as
red dots.
```

YIQ = rgb2ntsc(Lena);

This line converts the Lena image from RGB color space to the YIQ color space. The YIQ color space separates the image into its luminance (Y) and chrominance (I and Q) components.

Y = YIQ(:,:,1);

This line extracts the luminance component (Y) from the YIQ image.

laplacian_filtered = imfilter(Y, fspecial('laplacian'));

This line applies a Laplacian filter to the Y component of the image. The Laplacian filter enhances edges and fine details in the image.

gaussian_filtered = imgaussfilt(Y);

This line applies a Gaussian filter to the Y component of the image. The Gaussian filter smooths the image, reducing noise and blurring details.

sobel_filtered = edge(Y, 'Sobel');

This line applies the Sobel operator to the Y component of the image. The Sobel operator detects edges in the image.

harris_filtered = corner(Y);

This line applies the Harris corner detector to the Y component of the image. The Harris corner detector identifies corner points and points of interest in the image.

YIQ_filtered = cat(3, laplacian_filtered, YIQ(:,:,2), YIQ(:,:,3));

RGB_filtered = ntsc2rgb(YIQ_filtered);

These lines convert the filtered Y component back to RGB color space. The filtered Y component is combined with the original I and Q components to obtain the filtered RGB image.

grayLena = rgb2gray(Lena);

corners = detectHarrisFeatures(grayLena);

These lines convert the Lena image to grayscale and then apply the Harris corner detector to the grayscale image. The detectHarrisFeatures function identifies corner points in the image.

figure;

imshow(Lena);

title('Original Image');

hold on;

plot(corners.Location(:,1), corners.Location(:,2), 'r.', 'MarkerSize', 50);

hold off;

This code displays the original Lena image and overlays the corner points on top of it using red dots.