Student: Melis Kilic

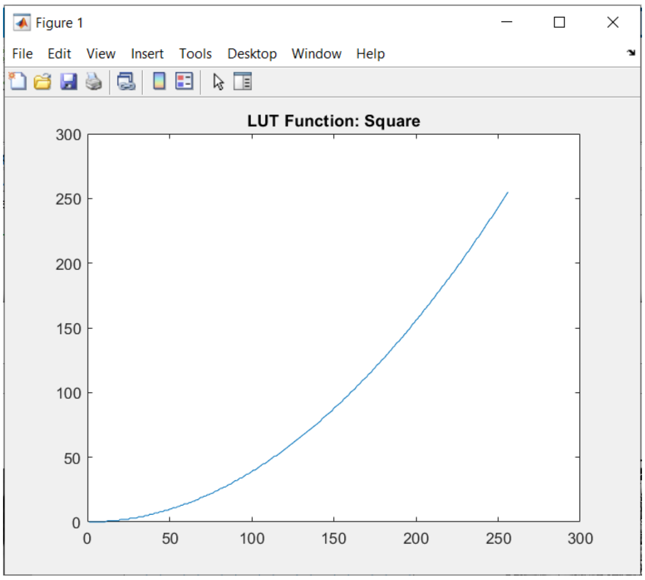
Basic operations on images

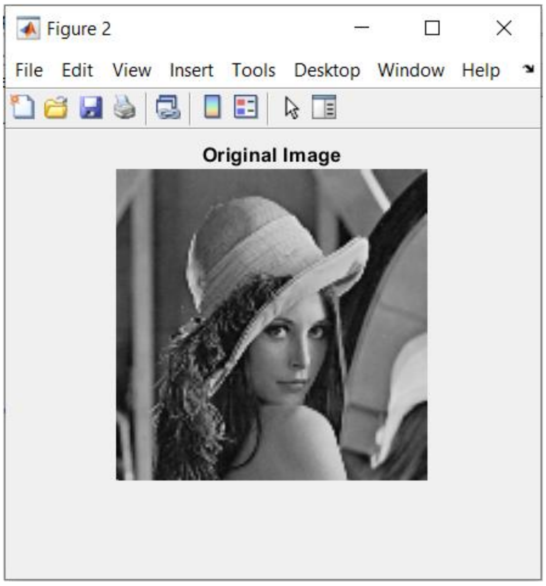
BO-1 LUT operation

```
clearvars;
close all;
clc;

% Load the provided "mat" file with predefined LUT functions.
load functionsLUT.mat;

% Display an exemplary function.
figure;
plot(square);
title('LUT Function: Square');

% Load a sample image and display it
image = imread('lena.bmp');
figure;
imshow(image);
title('Original Image');

% For a selected image perform LUT operations using at least three different LUT
function - for example use functions: square, inverse and saw. Display the result.
Then for each LUT create a figure with three subplots.

output_square = intlut(image, square);
output_inverse = intlut(image, inverse);
output_saw = intlut(image, saw);

% Figure for the "square" LUT function
figure('position', [100, 100, 1000, 300]);

subplot(1, 3, 1);
imshow(image);
title('Input Image');
daspect([1 1 1]);

subplot(1, 3, 2);
plot(square);
title('LUT Function: Square');
xlim([0 255]);
ylim([0 255]);
daspect([1 1 1]);

subplot(1, 3, 3);
imshow(output_square);
title('Output Image');
daspect([1 1 1]);

% Figure for the "inverse" LUT function
figure('position', [100, 100, 1000, 300]);

subplot(1, 3, 1);
imshow(image);
title('Input Image');
```

```
daspect([1 1 1]);

subplot(1, 3, 2);
plot(inverse);
title('LUT Function: Inverse');
xlim([0 255]);
ylim([0 255]);
daspect([1 1 1]);

subplot(1, 3, 3);
imshow(output_inverse);
title('Output Image');
daspect([1 1 1]);

% Figure for the "saw" LUT function
figure('position', [100, 100, 1000, 300]);

subplot(1, 3, 1);
imshow(image);
title('Input Image');
daspect([1 1 1]);

subplot(1, 3, 2);
plot(saw);
title('LUT Function: Saw');
xlim([0 255]);
ylim([0 255]);
daspect([1 1 1]);

subplot(1, 3, 3);
imshow(output_saw);
title('Output Image');
daspect([1 1 1]);
```

Result of the code:





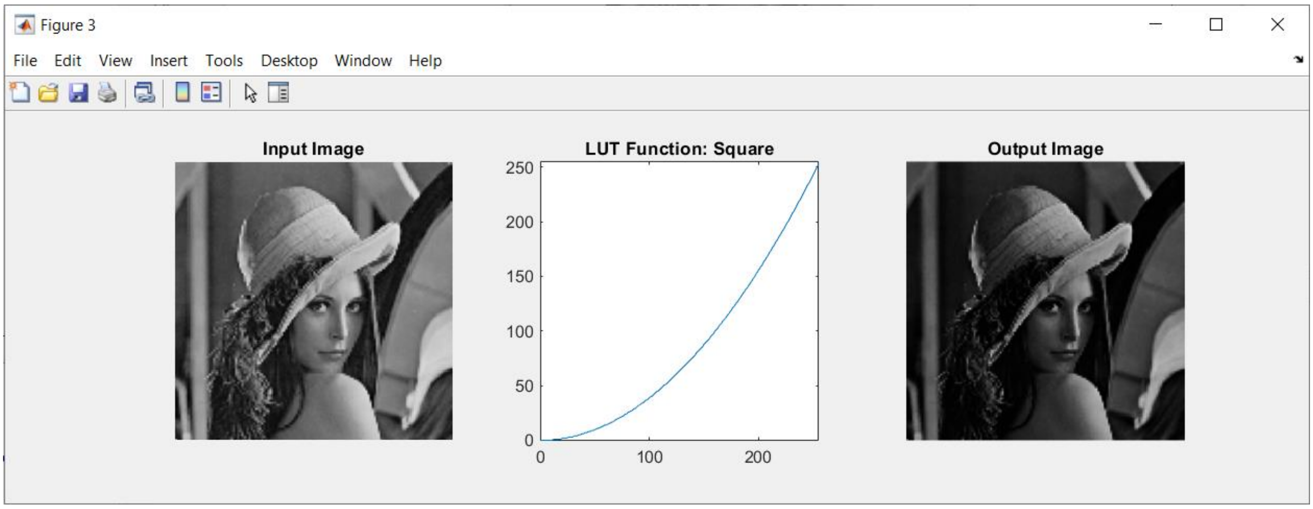


Figure 3
File Edit View Insert Tools Desktop Window Help
Input Image
LUT Function: Square
Output Image
250
200
150
100
50
0
0
100
200

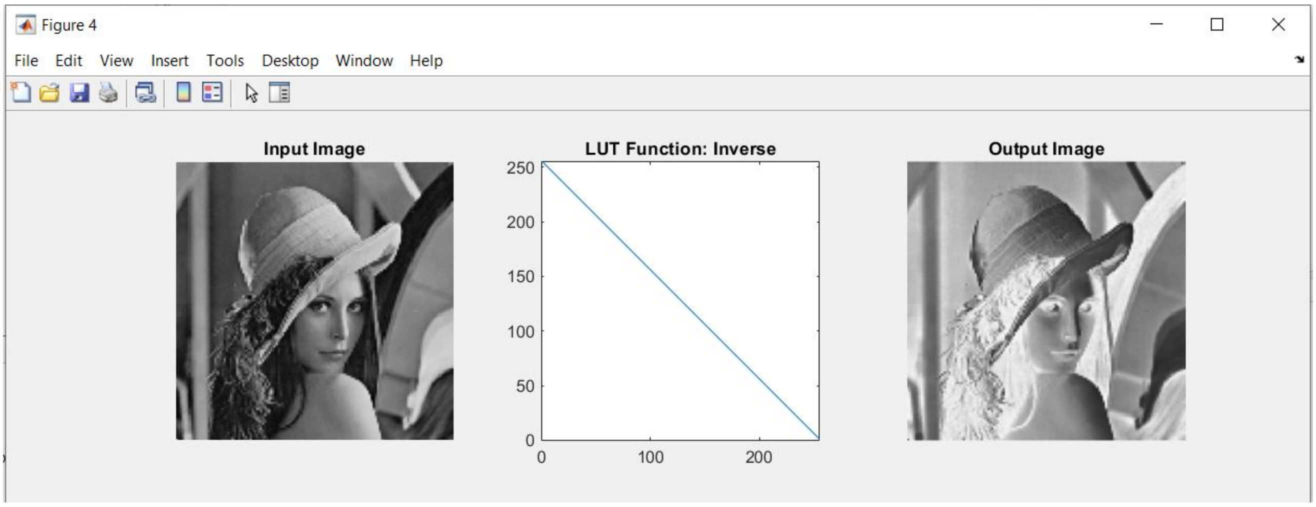


Figure 4
File Edit View Insert Tools Desktop Window Help
Input Image
LUT Function: Inverse
Output Image
250
200
150
100
50
0
0
100
200

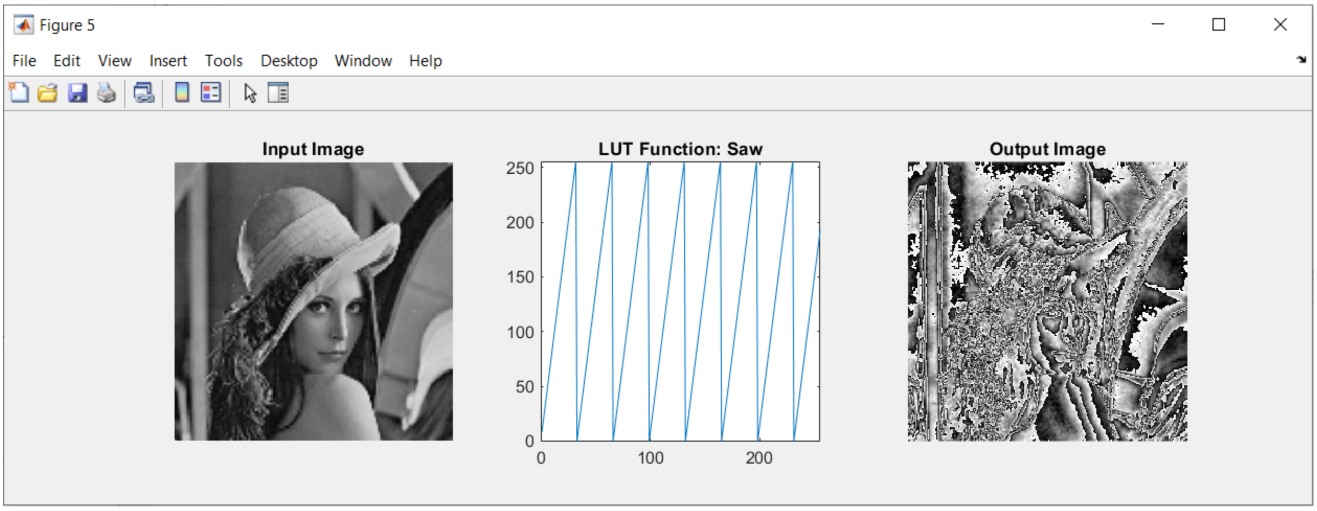


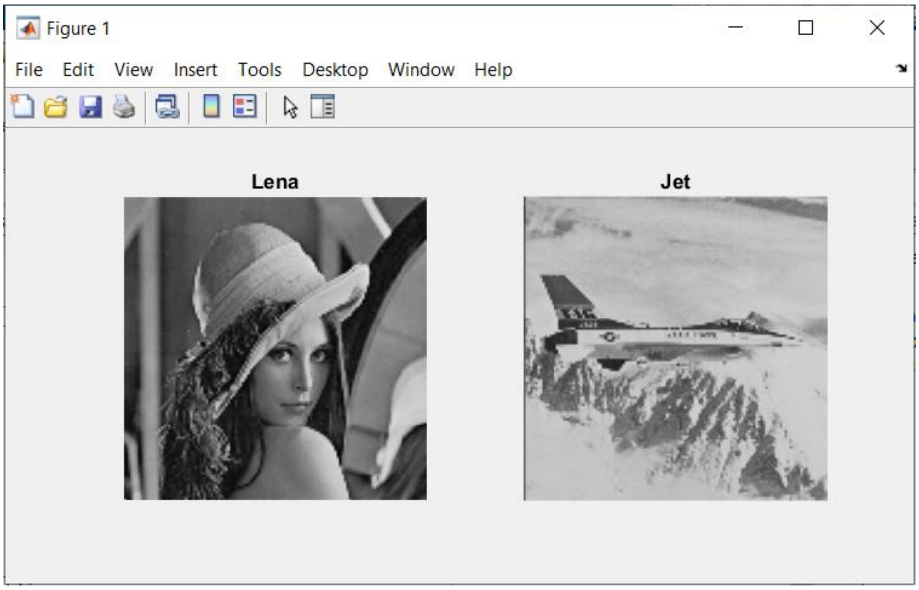
Figure 5
File Edit View Insert Tools Desktop Window Help
Input Image
LUT Function: Saw
Output Image
250
200
150
100
50
0
0
100
200

BO-2. Arithmetical operations

Addition

```
%%
clearvars;
close all;
clc;

% Load two images lena.bmp and jet.bmp and display them.

lena = imread('lena.bmp');
jet = imread('jet.bmp');

figure('position', [600, 450, 600, 300]);
subplot(1, 2, 1);
imshow(lena);
title('Lena');

subplot(1, 2, 2);
imshow(jet);
title('Jet');

% Add images Lena and Jet. Use the function C=imadd(X, Y). Display the result.

figure;
result = imadd(lena, jet);
imshow(result, []);
title('Image Addition Result');

% The imadd has the ability to specify the output data type. Try to use type
int16.

figure;
modified_result = imadd(lena, jet, 'int16');
imshow(modified_result, []);
title('Image Addition Modified Result (int16)');
```

Result of the code:



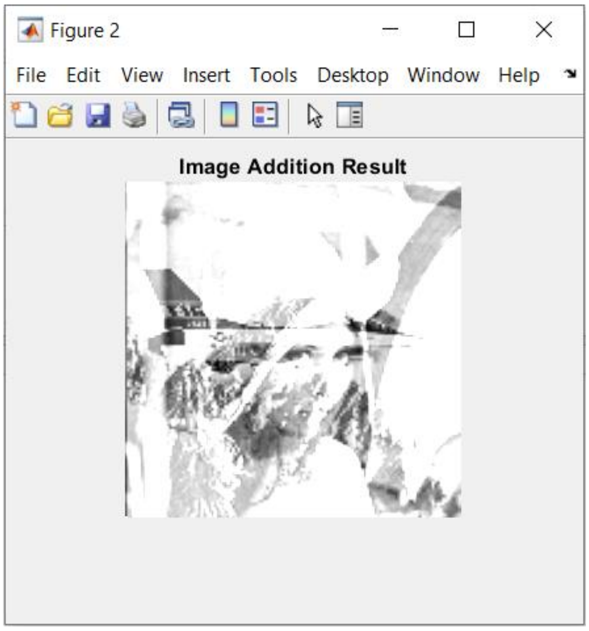


Image Addition Result

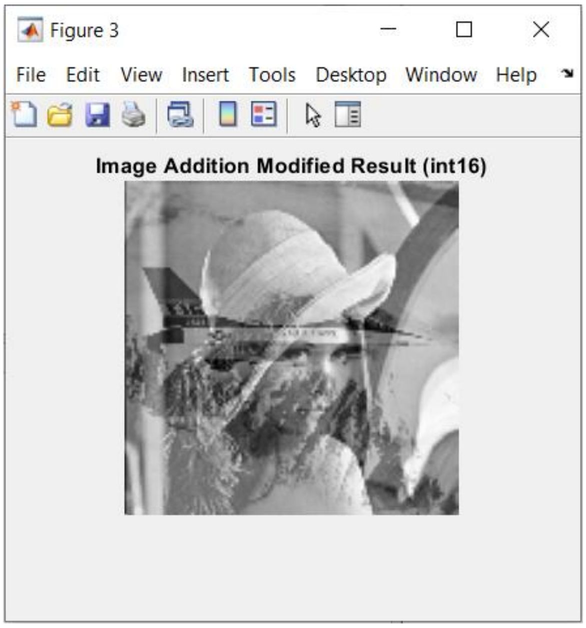
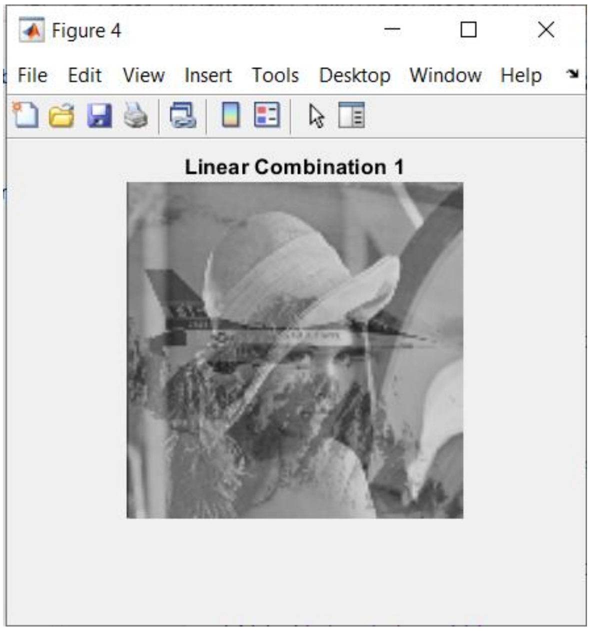


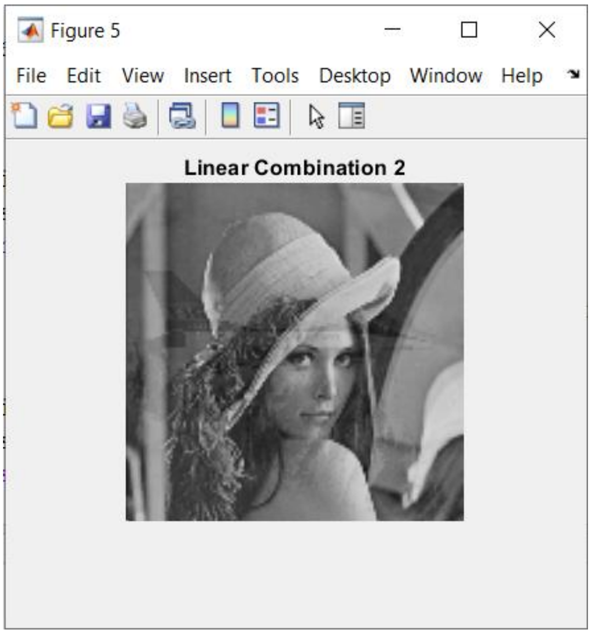
Image Addition Modified Result (int16)

Linear combination

```
% Coefficients for the linear combination

coeff_lena = 0.5;
coeff_jet = 0.5;

% Linear combination

linear_combination = imlincomb(coeff_lena, lena, coeff_jet, jet);
figure;
imshow(linear_combination);
title('Linear Combination 1');

coeff_lena = 0.8;
coeff_jet = 0.2;

linear_combination = imlincomb(coeff_lena, lena, coeff_jet, jet);

figure;
imshow(linear_combination);
title('Linear Combination 2');
```

Result of the code:

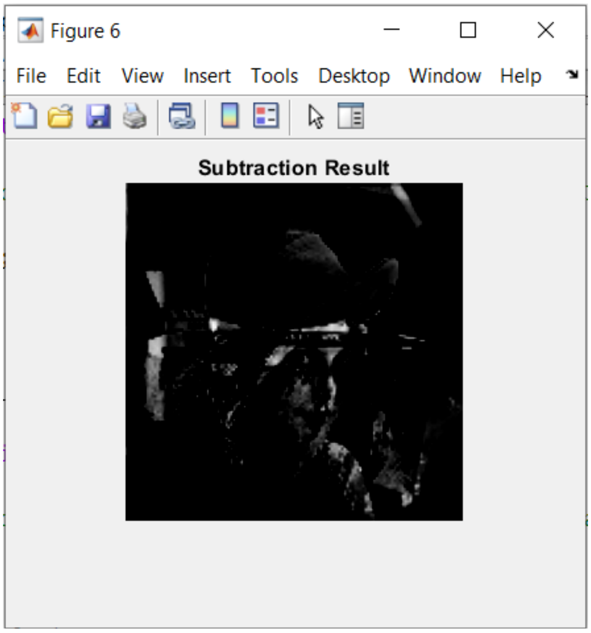




Subtraction

```
% Subtract images Lena and Jet. Use the function imsubtract.

figure;
result = imsubtract(lena, jet);
imshow(result, []);
title('Subtraction Result');

% To improve the result change the datatype for the images Lena and Jet from uint8
to int16.

lena_int16 = int16(lena);
jet_int16 = int16(jet);

figure;
result = imsubtract(lena_int16, jet_int16);
imshow(result, []);
title('Subtraction Modified Result (int16)');

% Use the function imabsdiff to obtain the absolute difference of images Lena and
Jet.

figure;
result = imabsdiff(lena, jet);
imshow(result, []);
title('Absolute Difference Result');
```

Result of the code:



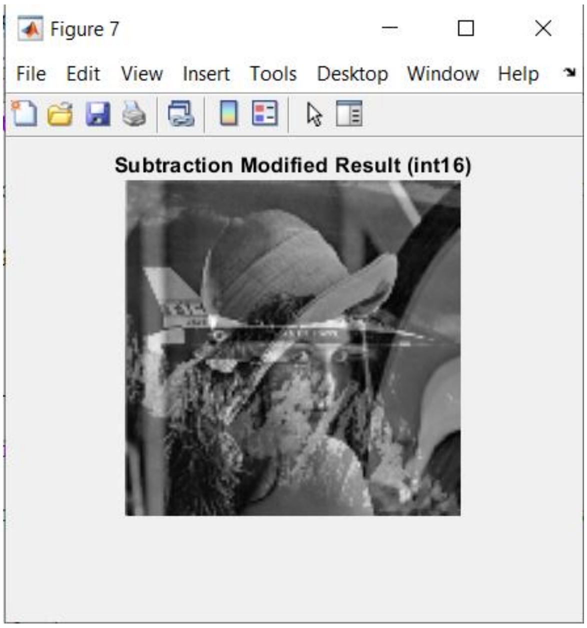


Figure 7
File Edit View Insert Tools Desktop Window Help
Subtraction Modified Result (int16)

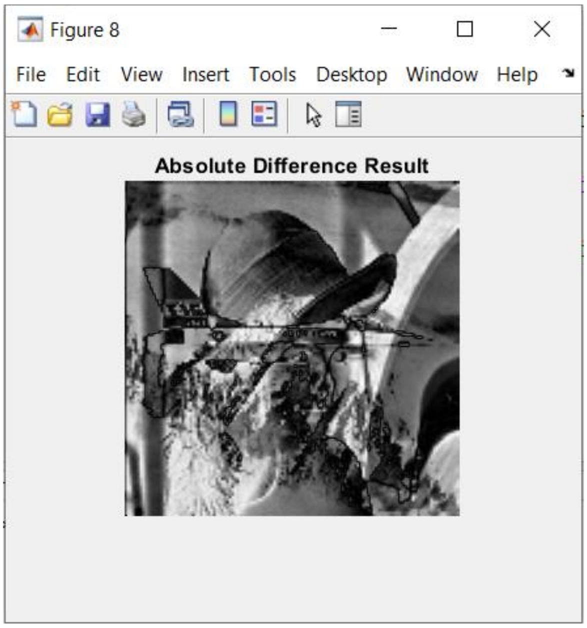


Figure 8
File Edit View Insert Tools Desktop Window Help
Absolute Difference Result

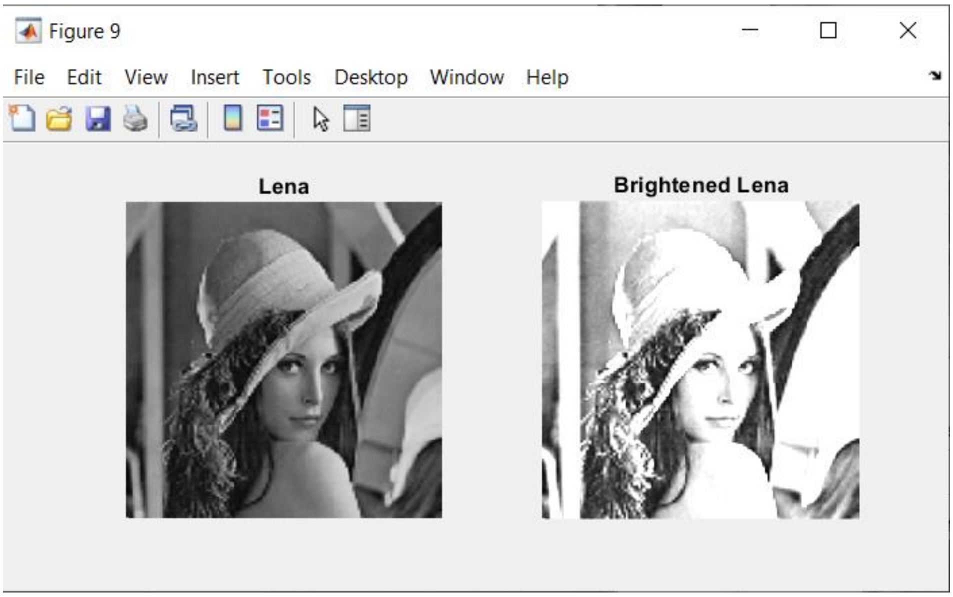
Is the result of the subtraction satisfactory? What could adversely affect the outcome of the operation? To improve the result change the datatype for the images Lena and Jet from uint8 to int16. Why this change improves the result?

Changing the data type to int16 improves the result because int16 is signed and can represent both positive and negative values.

Multiplication

```
% Constant factor for multiplication
constant_factor = 2.5;

% Multiplication by the constant factor
brightened_image = immultiply(lena, constant_factor);

figure;
subplot(1,2,1);
imshow(lena);
title('Lena');

subplot(1,2,2);
imshow(brightened_image, []);
title('Brightened Lena');
```

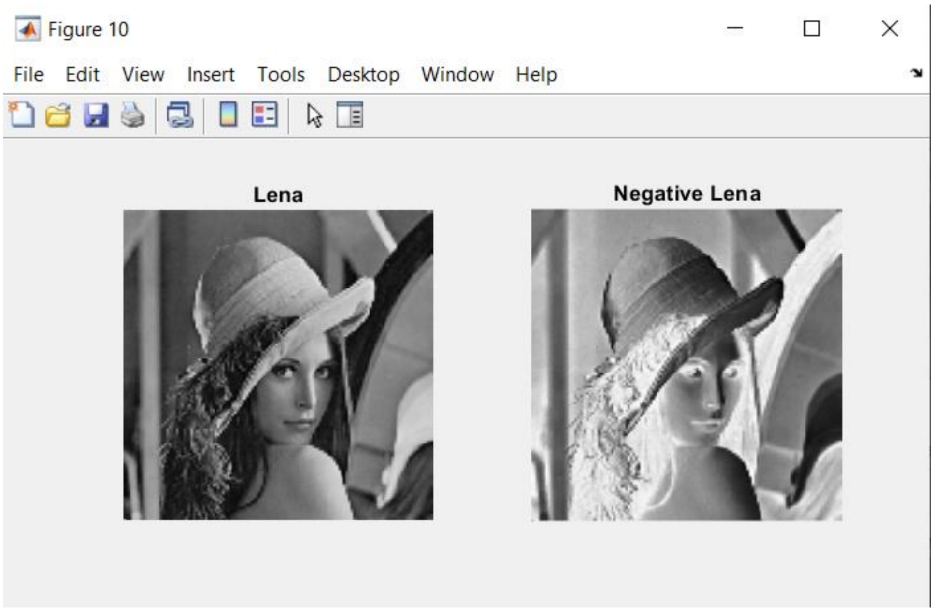
Result of the code:



Negation

```
figure;
subplot(1,2,1);
imshow(lena);
title('Lena');

% Use the imcomplement function on a selected image and display the result.
negative_lena = imcomplement(lena);

subplot(1,2,2);
imshow(negative_lena, []);
title('Negative Lena');
```

Result of the code:



BO-3. Logical operations

```
%%
clearvars;
close all;
clc;

% Load two images circle.bmp and square.bmp. Convert the loaded images to logical
type: circle = logical(circle). View the loaded images.

circle = imread('circle.bmp');
square = imread('square.bmp');

circle = logical(circle);
square = logical(square);

figure;
subplot(1, 2, 1);
imshow(circle);
title('Circle Image (Logical)');

subplot(1, 2, 2);
imshow(square);
title('Square Image (Logical)');

% On the loaded images perform selected logical operations:
NOT (~), AND (&), OR (|), XOR (xor). Display the results in 4 (2×2) subplots.

not_result = ~circle;
and_result = circle & square;
or_result = circle | square;
xor_result = xor(circle, square);
```

```
figure;

subplot(2, 2, 1);
imshow(not_result);
title('NOT Operation');

subplot(2, 2, 2);
imshow(and_result);
title('AND Operation');

subplot(2, 2, 3);
imshow(or_result);
title('OR Operation');

subplot(2, 2, 4);
imshow(xor_result);
title('XOR Operation');
```

Result of the code:

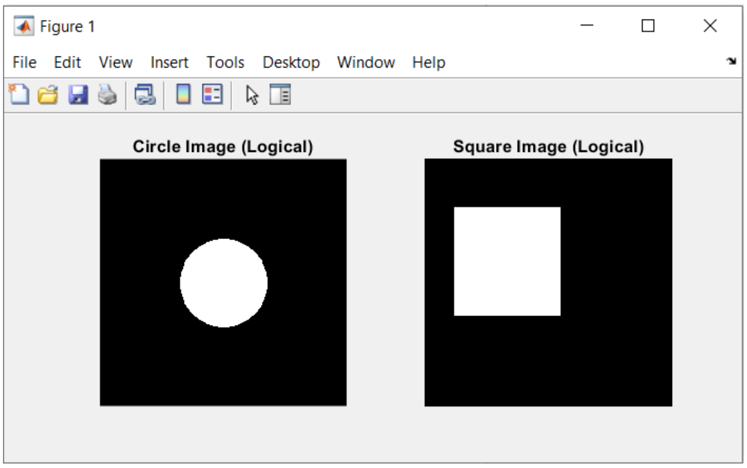


Figure 1
File Edit View Insert Tools Desktop Window Help
Circle Image (Logical)
Square Image (Logical)

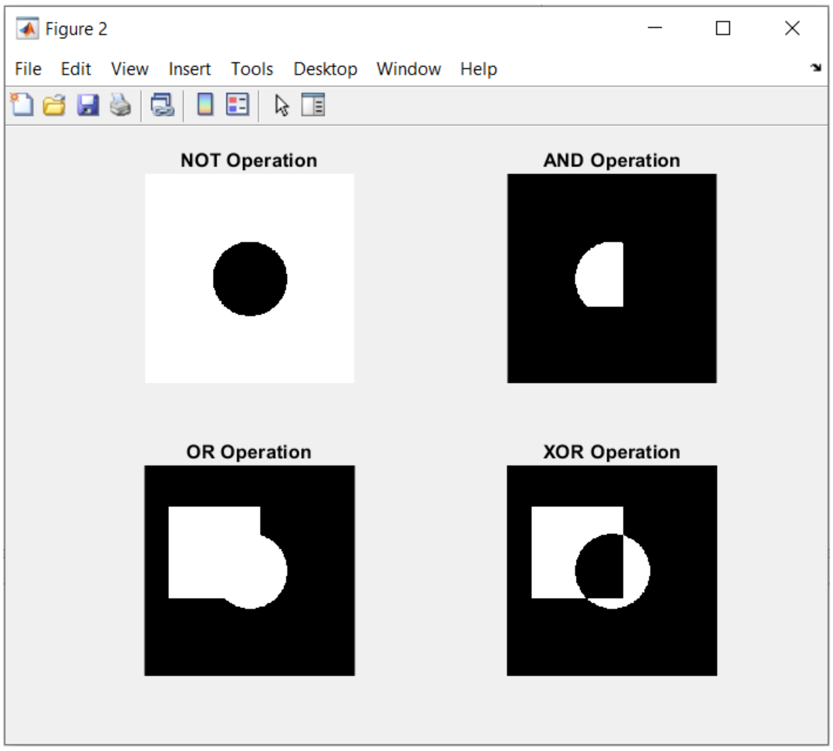


Figure 2
File Edit View Insert Tools Desktop Window Help
NOT Operation
AND Operation
OR Operation
XOR Operation