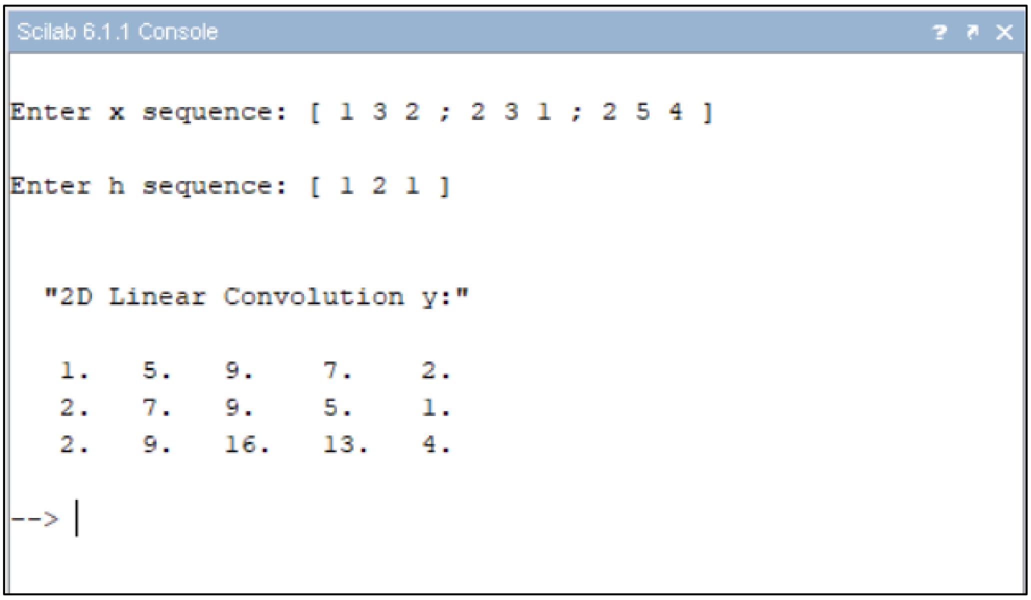
Practical 01

Aim: 2D Linear Convolution, Circular Convolution between two 2D matrices.

Code and Output:

A. 2D Linear Convolution

```
clc;
clear all;
close;
x = input('Enter x sequence: ');
h = input('Enter h sequence: ');
lc = conv2(x,h)//Linear Convolution
disp('2D Linear Convolution y:', lc);
```



```
Scilab 6.1.1 Console

Enter x sequence: [ 1 3 2 ; 2 3 1 ; 2 5 4 ]

Enter h sequence: [ 1 2 1 ]


  "2D Linear Convolution y:"

   1.   5.   9.    7.    2.
   2.   7.   9.    5.    1.
   2.   9.   16.   13.   4.

--> 
```

B. 2D Circular Convolution

```
clc;
clear all;
close;
x = input('Enter x sequence: ');
h = input('Enter h sequence: ');

X = fft2 (x); // 2D FFT of x matrix
H = fft2 (h); // 2D FFT of h matrix
Y = X.*H; // Element by Element multiplication
```

```
y = fft2 (Y);
disp ('2D Circular Convolution y:', y);
```

```
Scilab 6.1.1 Console

Enter x sequence: [ 1 2 2 ; 1  0 1 ; 1 2 1 ]

Enter h sequence: [ 1 2 1 ; 1  1 1 ; 2 1 0 ]


  "2D Circular Convolution y:"

   126.   117.   99.
   99.    126.   108.
   99.    99.    117.

--> 
```

Practical 02

Aim: Circular Convolution expressed as linear convolution plus alias.

Code and Output:

```
clc;
clear all;
close;
x = input('Enter x sequence: ');
h = input('Enter h sequence: ');
y = conv2(x,h); //Linear Convolution
y1 = [y(:,1)+y(:,$),y(:,2)];
y2 = [y1(1,:)+y1($,:);y1(2,:)] //Circular
disp('linear Convolution result y:', y);
disp('Circular Convolution Expressed as Linear Convolution plus alias:', y2);
```

Scilab 6.1.1 Console

```
Enter x sequence: [ 2 3 ; 4 1 ]

Enter h sequence: [ 1 2 ; 2 1 ]


  "linear Convolution result y:"

   2.   7.    6.
   8.   17.   5.
   8.   6.    1.

  "Circular Convolution Expressed as Linear Convolution plus alias:"

   17.   13.
   13.   17.

-->
```

Practical 03

Aim: Linear Cross correlation of a 2D matrix, Circular correlation between two signals and Linear auto correlation of a 2D matrix.

Code and Output:

A. Linear Cross correlation of a 2D matrix

```
clc;
clear all;
close;
x = input('Enter x sequence: ');
h1 = input('Enter h sequence: ');
h2 = h1(:,$:-1:1);
h = h2($:-1:1,:);
y = conv2(x,h);
disp(' Linear Cross Correlation y:', y);
```

```
Scilab 6.1.1 Console

Enter x sequence: [ 1 2 ; 3 1 ]

Enter h sequence: [ 1 4 ; 5 2 ]


  " Linear Cross Correlation y:"

   2.    9.    10.
   10.   26.   7.
   12.   7.    1.

-->
```

B. Circular correlation between two signals

```
clc;
clear all;
close;
x = input('Enter x sequence: ');
h = input('Enter h sequence: ');
h = h(:,$:-1:1);
h = h($:-1:1,:);
```

```
X = fft2(x);
H = fft2(h);
Y = X.*H;
y = fft2(Y);
disp ('Circular Correlation y:', y);
```

```
Scilab 6.1.1 Console

Enter x sequence: [ 4 2 ; 3 2 ]

Enter h sequence: [ 1 4 ; 5 2 ]


  "Circular Correlation y:"

   128.   140.
   136.   124.

-->
```

C. Linear auto correlation of a 2D matrix

```
clc;
clear all;
close;
x1 = input('Enter x sequence: ');
x2 = x1(:,$:-1:1) ;
x2 = x2($:-1:1,:) ;
x = conv2(x1, x2);
disp ('Linear Auto Correlation x:', x);
```

```
Scilab 6.1.1 Console

Enter x sequence: [ 1 2 1 ; 1 2 2 ; 2 1 2 ]


  "Linear Auto Correlation x:"

   2.    5.    6.    5.    2.
   4.    11.   15.   10.   5.
   7.    14.   24.   14.   7.
   5.    10.   15.   11.   4.
   2.    5.    6.    5.    2.

-->
```

Practical 04

Aim: DFT of 4x4 gray scale image.

Code and Output:

```
clc;
clear all;
close;
img = input('4x4 grayscale image: ');
F = fft(img);
disp('2D Discrete Fourier Transformed 2D Image: ', F);
```

```
Scilab 6.1.1 Console

4x4 grayscale image: [ 1 1 1 1 ; 1 1 1 1 ; 1 1 1 1 ; 1 1 1 1 ]

  "2D Discrete Fourier Transformed 2D Image: "

   16.   0.   0.   0.
   0.    0.   0.   0.
   0.    0.   0.   0.
   0.    0.   0.   0.

-->
```

Practical 05

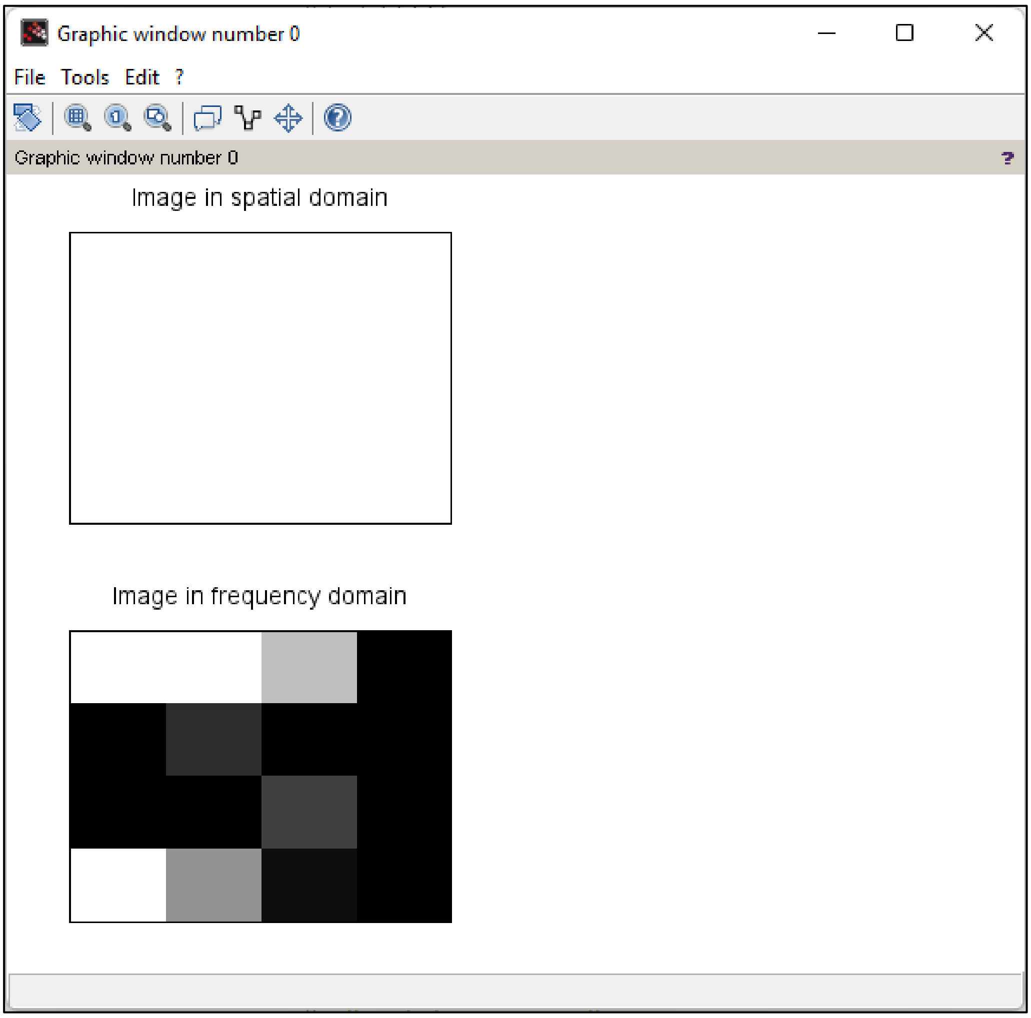
Aim: Compute discrete cosine transform, Program to perform KL transform for the given 2D matrix.

Code and Output:

A. Compute discrete cosine transform

```
clc;
clear all;
close;

f = input("Enter your sequence:");
[M,N] = size(f);
const=sqrt(2/N);
for k=0:1:N-1
   for l=0:1:N-1
      if k==0
         c(k+1,l+1)=1/sqrt(N);
      else
         a=2*l;
         c(k+1,l+1)=const*cos((%pi*k*(a+1))/(2*N));
      end
   end
end

f1=c*f;
disp(f1);
F=c*f*c';
disp('Discrete Cosine Transfor', c);
disp('Discrete Cosine Transform of f(m,n) is', F);
subplot(221);
imshow(f);
title('Image in spatial domain')
subplot(223);
imshow(F);
title('Image in frequency domain')
```

```
Scilab 6.1.1 Console

Enter your sequence:[ 2 3 1 2 ; 4 3 2 1 ; 6 4 3 2 ; 2 1 1 2 ]


   7.           5.5          3.5          3.5
  -0.5411961   1.0359649  -0.2705981  -0.2705981
  -3.          -1.5         -1.5         0.5
   1.306563    1.1944776   0.6532815   0.6532815

  "Discrete Cosine Transfor"

   0.5          0.5          0.5          0.5
   0.6532815   0.2705981  -0.2705981  -0.6532815
   0.5         -0.5         -0.5          0.5
   0.2705981  -0.6532815   0.6532815  -0.2705981

  "Discrete Cosine Transform of f(m,n) is"

   9.75         2.8276813   0.75        -0.3594698
  -0.0232136   0.1767767  -0.7885805  -0.9267767
  -2.75        -2.2864852   0.25        -0.9470932
   1.9038018   0.5732233   0.0560427  -0.1767767

-->
```



B. Program to perform KL transform for the given 2D matrix

```
clc;
clear all;
close;

X = input('Enter a 3*4 Sequence: ')  //3*4
[m,n]= size(X);                   //m=3 ,  n=4
A = [0];
E = [0];
for i =1 : n
  A = A+X(:,i);
  E = E+X(:,i)*X(:,i)';
end

mx = A/n; //mean matrix
E = E/n;
C = E - mx*mx'; // covariance matrix C = E[xx'] - mx*mx'
[V,D] = spec(C); // eigen values and eigen vectors
d = diag(D); // diagonal elements of eigen values
[d,i] = gsort(d); // sorting the elements of D in descending order
for j = 1: length (d)
 T(:,j)= V(:,i(j));
end
T=T'
disp ('Eigen Values are U = ', d)
disp ('The eigen vector matrix T =', T)
disp ('The KL tranform basis is =', T)
//KL transform
for i=1:n
Y(:,i)=T*X(:,i);
end
disp ('KL transformation of the input matrix Y =', Y)
// Reconstruction
for i=1:n
  x(:,i)= T '*Y(:,i);
end
disp ('Reconstruct matrix of the given sample matrix X =' ,x)
```

Scilab 6.1.1 Console

```
Enter a 3*4 Sequence: [ 2 1 3 1 ; 3 4 2 1 ; 2 6 4 1 ]


  "Eigen Values are U = "

   4.5000000
   0.7500000
   0.3750000

  "The eigen vector matrix T ="

   1.937D-18   0.4472136   0.8944272
   0.9128709  -0.3651484   0.1825742
   0.4082483   0.8164966  -0.4082483

  "The KL tranform basis is ="

   1.937D-18   0.4472136   0.8944272
   0.9128709  -0.3651484   0.1825742
   0.4082483   0.8164966  -0.4082483

  "KL transformation of the input matrix Y ="

   3.1304952   7.1554175   4.472136    1.3416408
   1.0954451   0.5477226   2.7386128   0.7302967
   2.4494897   1.2247449   1.2247449   0.8164966

  "Reconstruct matrix of the given sample matrix X ="

   2.   1.   3.   1.
   3.   4.   2.   1.
   2.   6.   4.   1.

-->
```

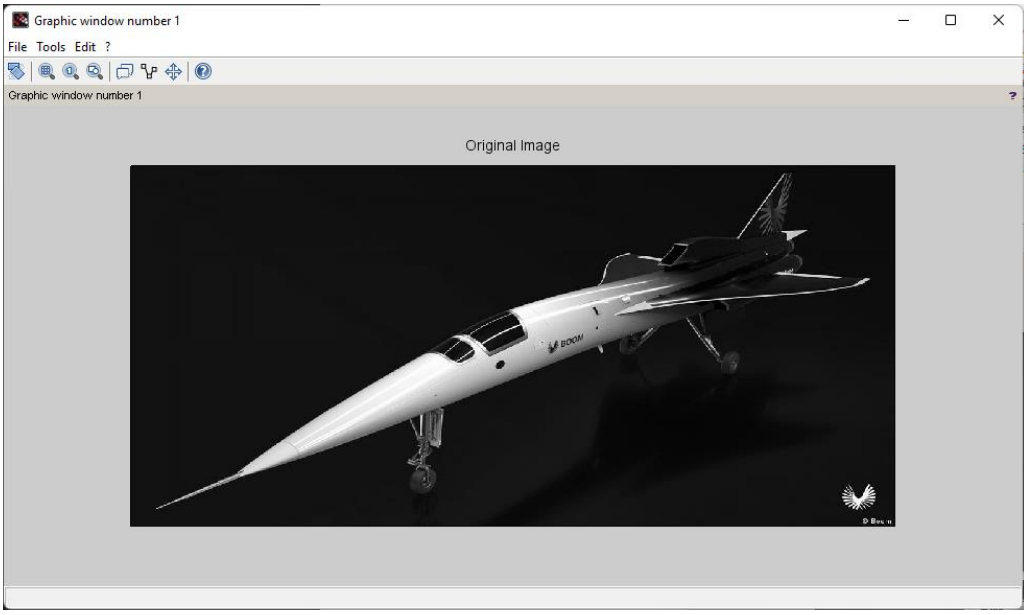
Practical 06

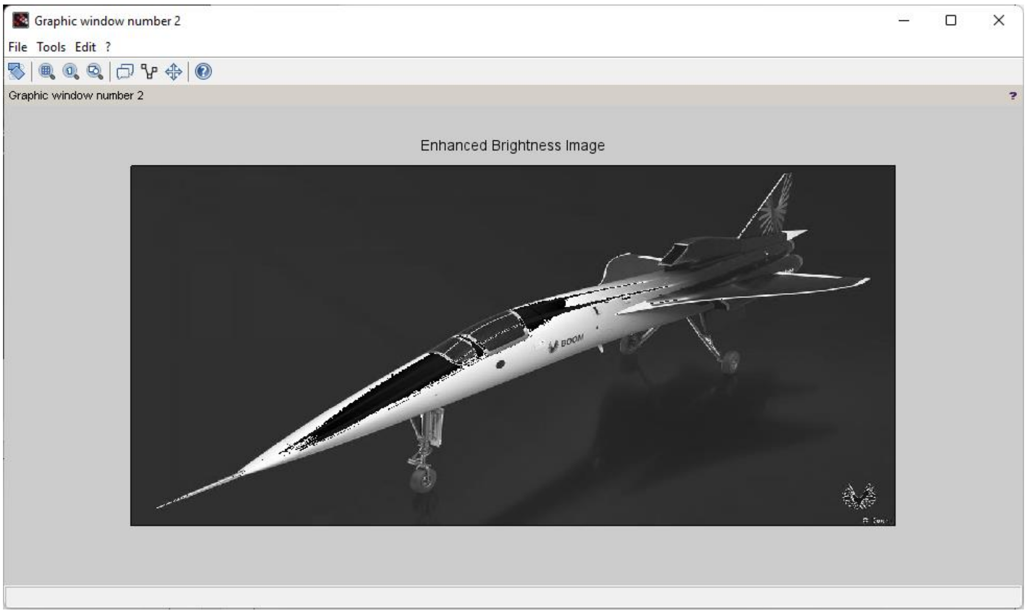
Aim: Brightness enhancement of an image, Contrast Manipulation, Image Negative.

Code and Output:

A. Brightness Enhancement

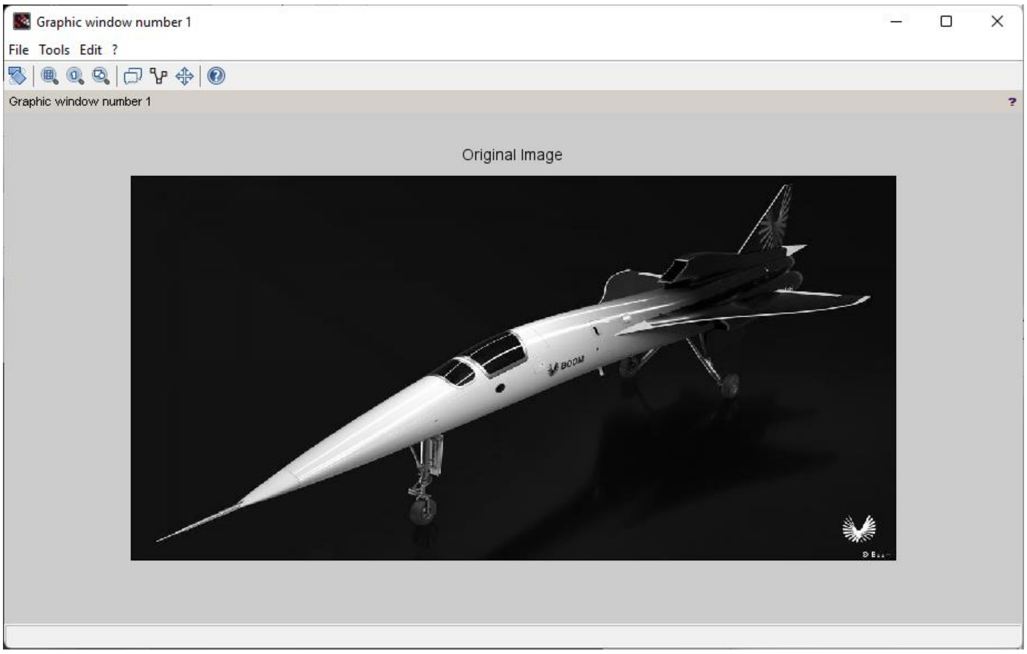
```
clc;
clear all;
close;
a=imread("p6.jpg");
a=rgb2gray(a);
b=double(a)+30;
b1=uint8(b)
figure(1)
imshow(uint8(a))
title("Original Image")
figure(2)
imshow(uint8(b))
title("Enhanced Brightness Image")
```

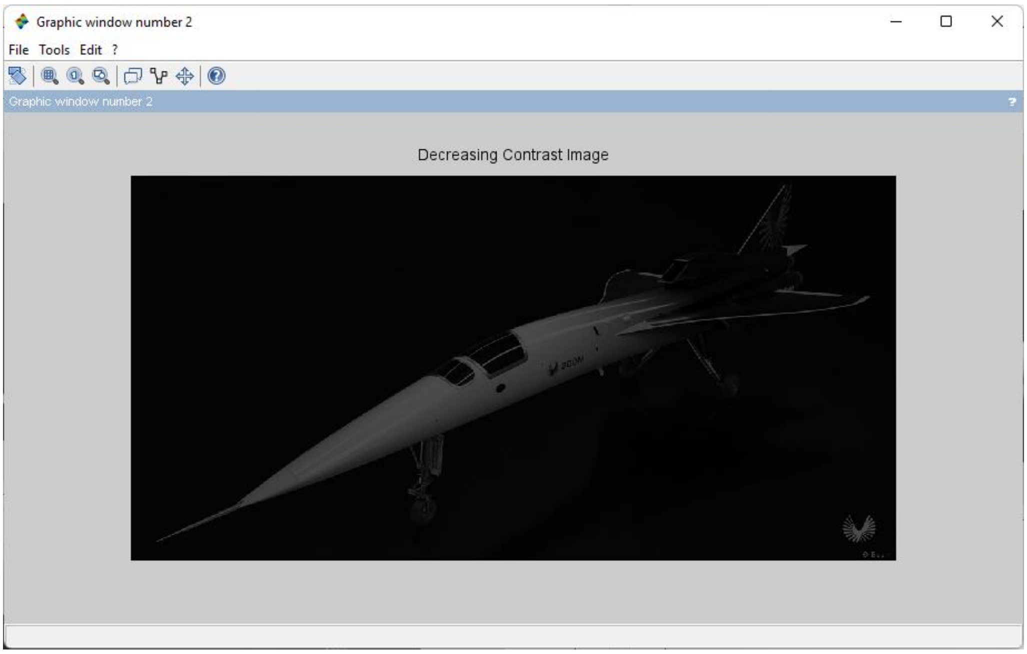


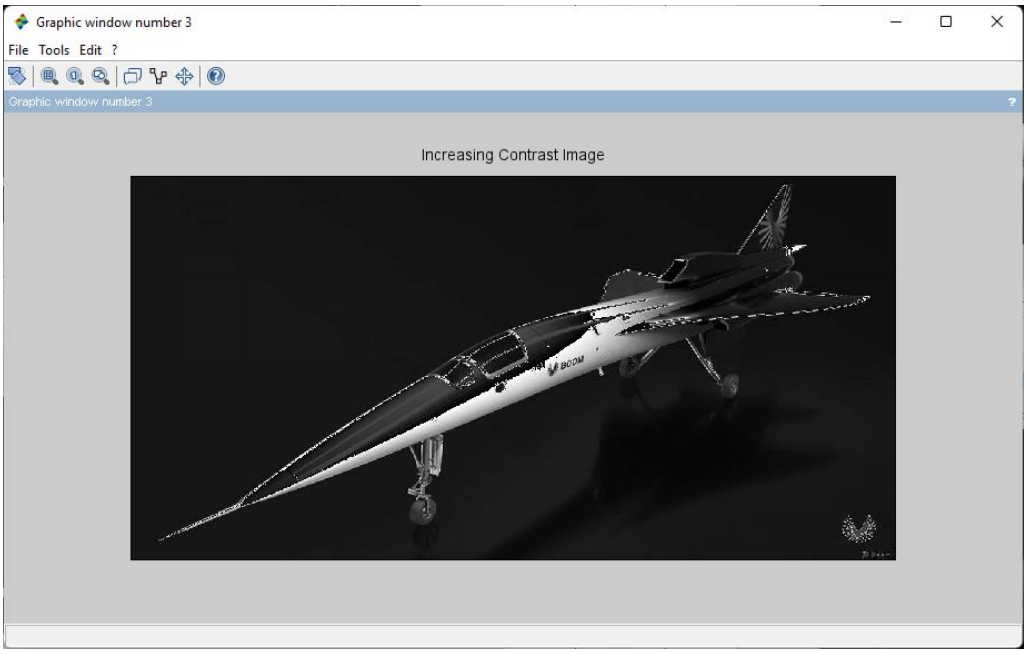


B. Contrast Manipulation

```
clc;
clear all;
close;
a=imread("p6.jpg");
a=rgb2gray(a);
b=double(a)*0.3;
b1=uint8(b)
c=double(a)*1.3;
c1=uint8(c)
figure(1)
imshow(uint8(a))
title("Original Image")
figure(2)
imshow(uint8(b))
title(" Decreasing Contrast Image")
figure(3)
imshow(uint8(c))
title(" Increasing Contrast Image")
```

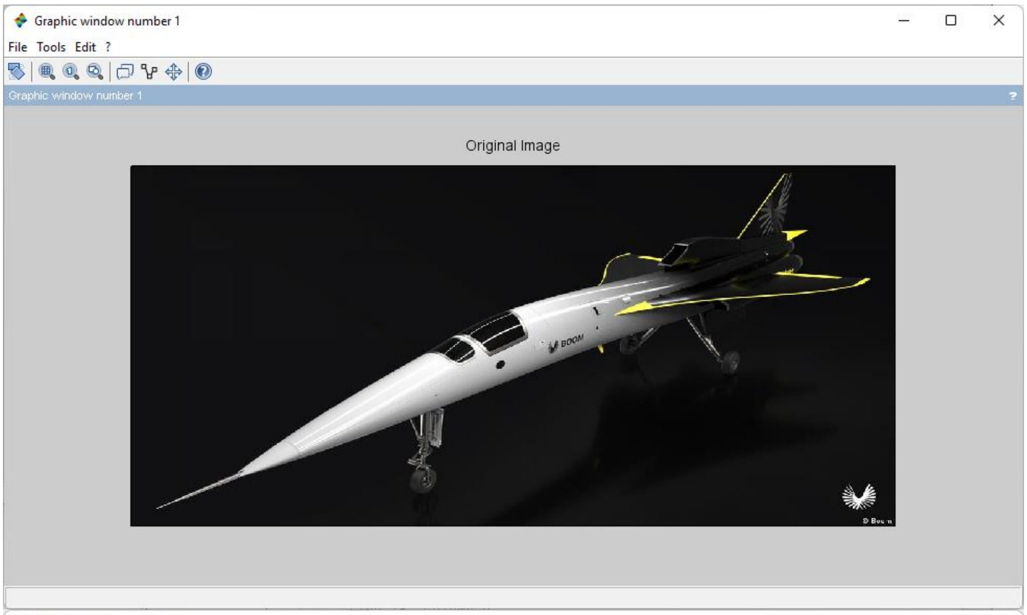

Graphic window number 1
File Tools Edit ?
Graphic window number 1
Original Image

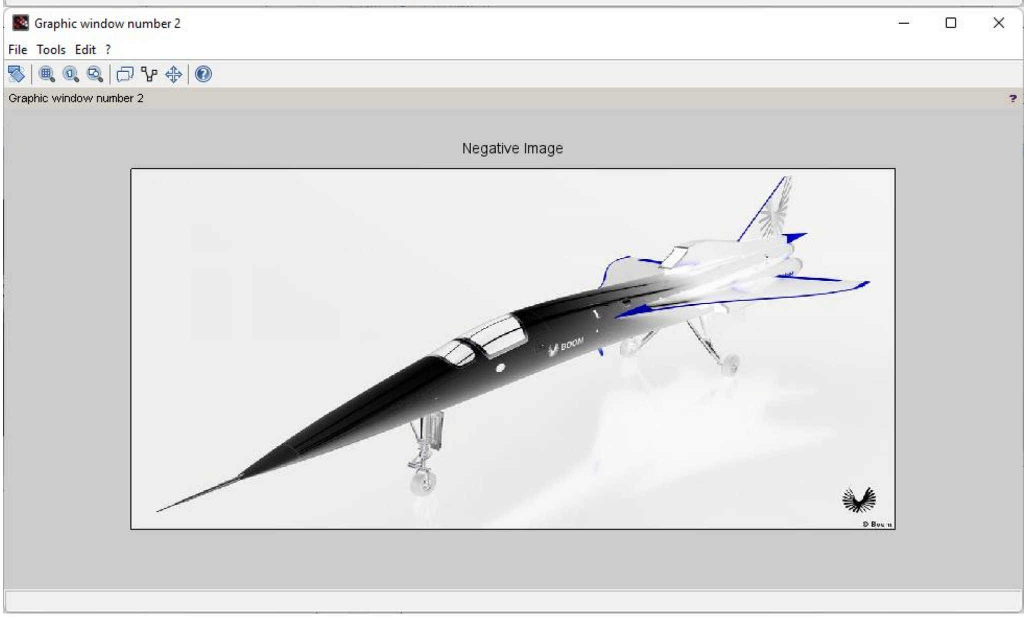

Graphic window number 2
File Tools Edit ?
Graphic window number 2
Decreasing Contrast Image



C. Image Negative

```
clc;
clear all;
close;
a=imread("p6.jpg");
b=255-double(a);
b1=uint8(b)
figure(1)
imshow(uint8(a))
title("Original Image")
figure(2)
imshow(uint8(b))
title("Negative Image")
```


Graphic window number 1
File Tools Edit ?
Graphic window number 1
Original Image
BOOM


Graphic window number 2
File Tools Edit ?
Graphic window number 2
Negative Image
BOOM

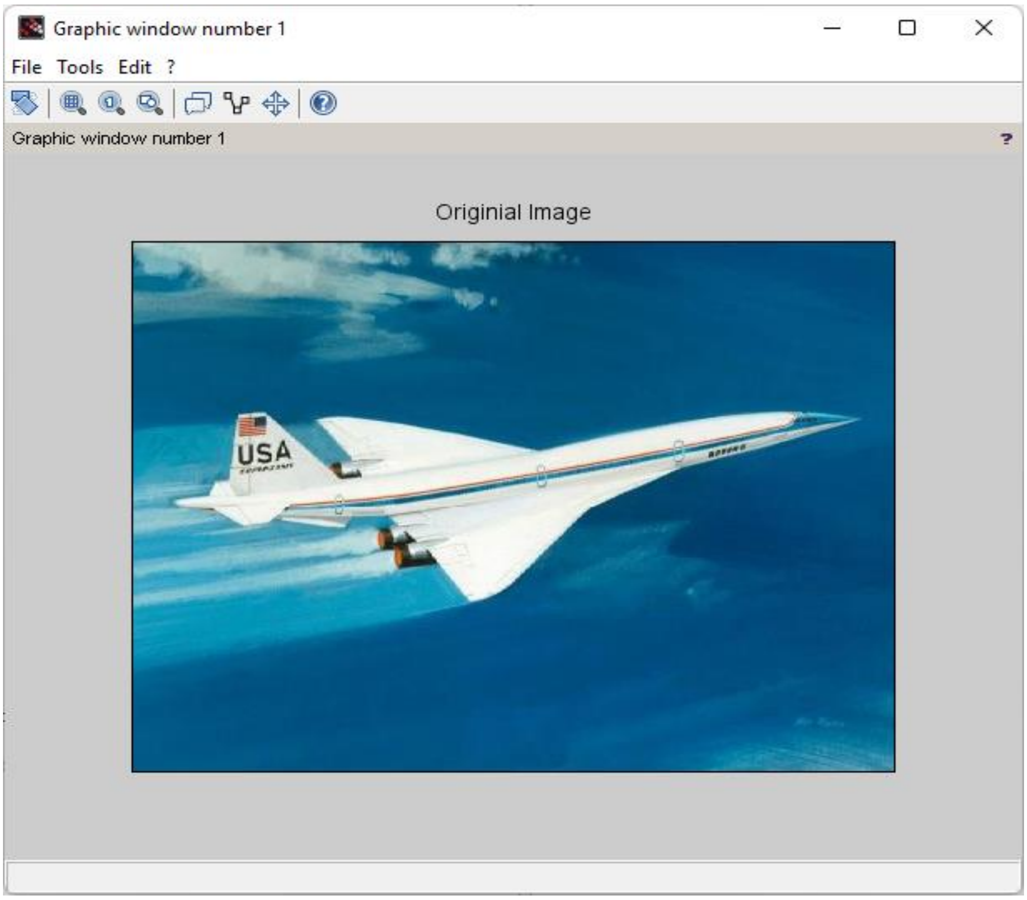
Practical 07

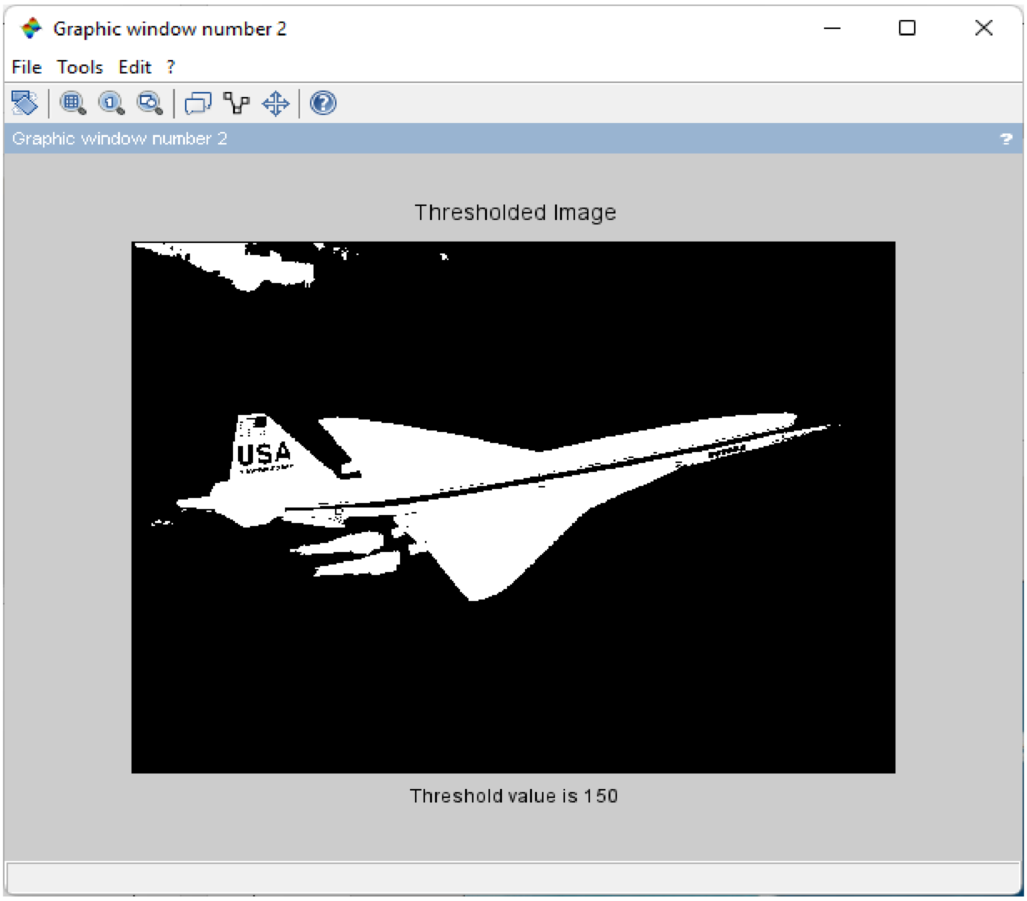
Aim: Perform threshold operation, perform gray level slicing without background.

Code and Output:

A. Perform threshold operation

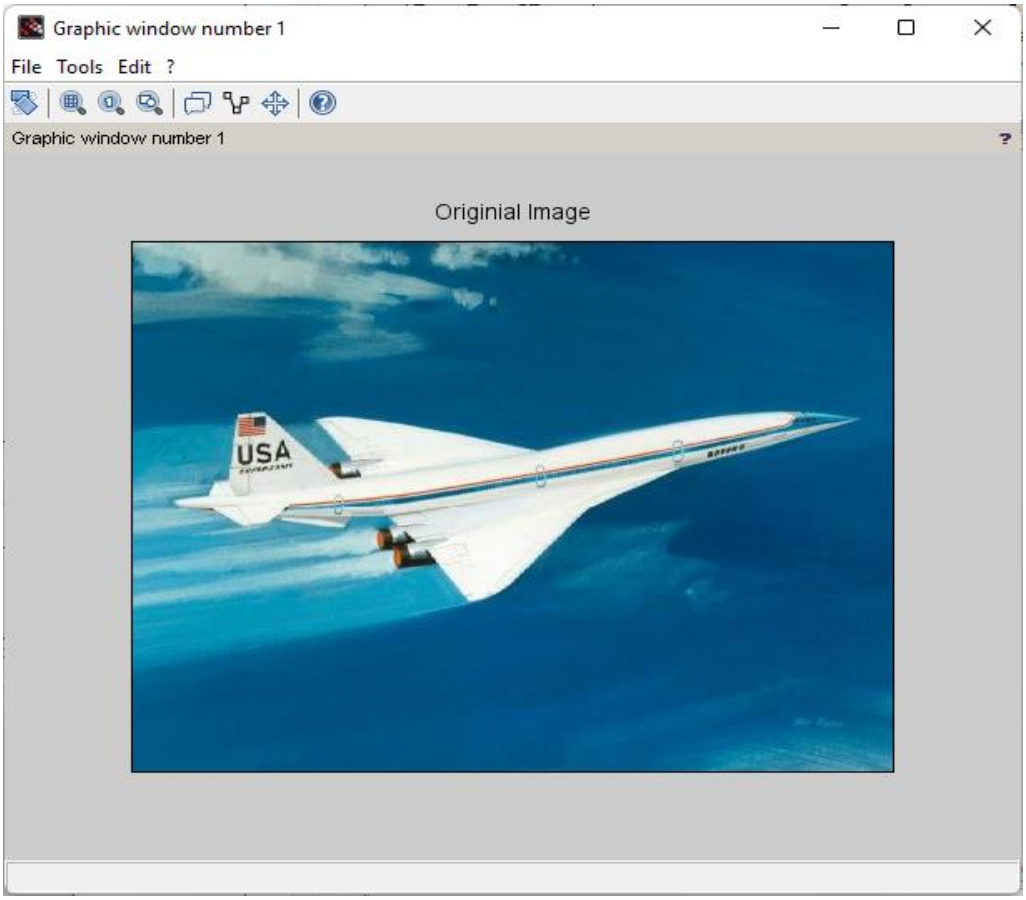
```
clc;
clear all;
close;

a = imread('p7.jpg');
a = double(a);
[m n] = size(a);
t = input ('Enter the Threshold Parameter: ');
for i = 1:m
  for j = 1:n
    if(a(i,j)<t)
      b(i,j)=0;
    else
      b(i,j)=255;
    end
   end
end
figure (1)
imshow (uint8(a));
title ('Originial Image' )
figure (2)
imshow (uint8(b));
title ('Thresholded Image')
xlabel (sprintf('Threshold value is %g', t))
```

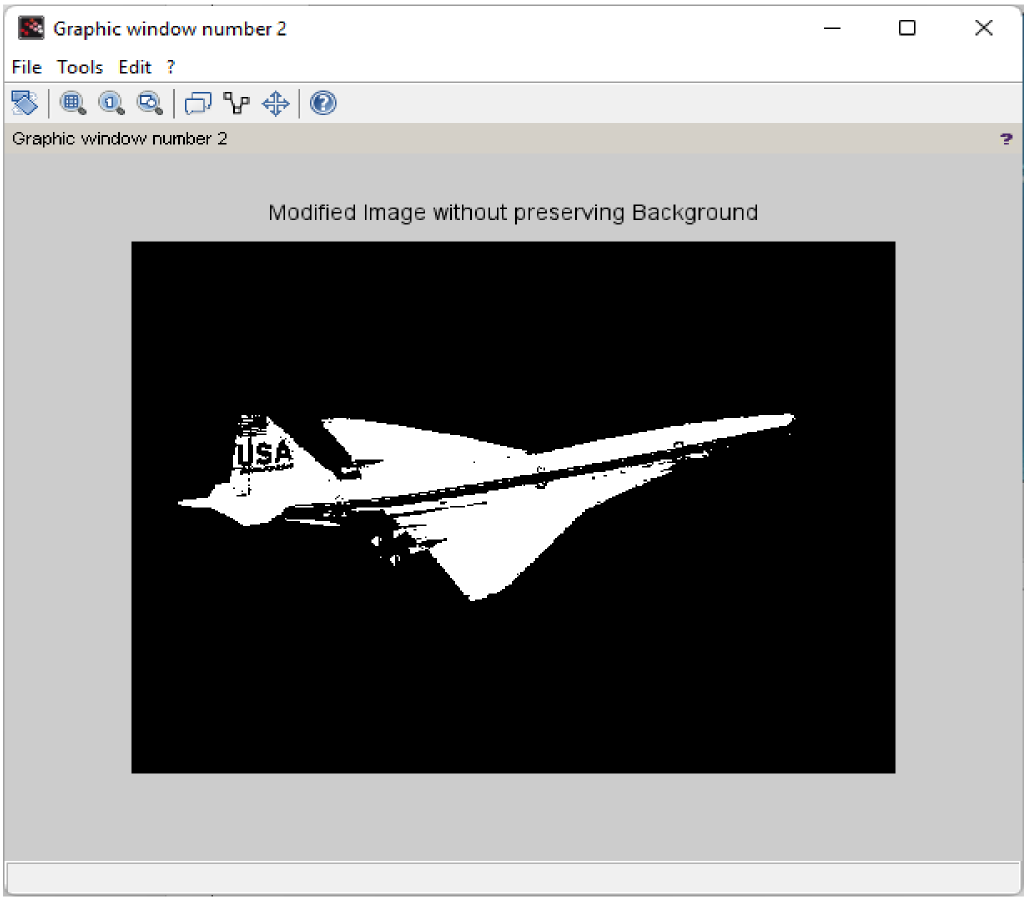

Graphic window number 1
File Tools Edit ?
Graphic window number 1
Originial Image
USA


Graphic window number 2
File Tools Edit ?
Graphic window number 2
Thresholded Image
USA
Threshold value is 150

B. Gray level slicing without background

```
clc;
clear all;
close;

x = imread('p7.jpg');

y = double(x);
[m n] = size(y);
L=double(255);

a=double(round(L/1.25));
b=double(round(2*L/2));

for i = 1:m
   for j = 1:n
      if(y(i,j)>= a & y(i,j) <=b)
         z(i,j)=L;
      else
         z(i,j)=0;
      end
    end
end
figure (1)
imshow (uint8(y));
title ('Originial Image' )
figure (2)
imshow (uint8(z));
title ('Modified Image without preserving Background')
```


Graphic window number 1
File Tools Edit ?
Graphic window number 1
Originial Image
USA

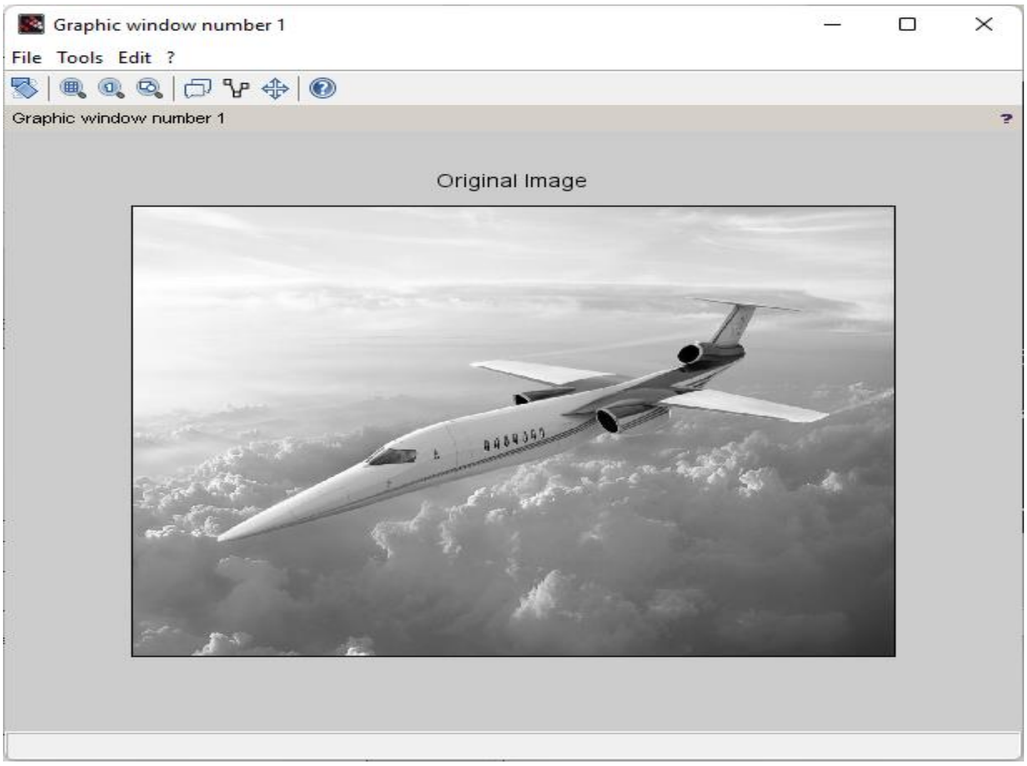

Graphic window number 2
File Tools Edit ?
Graphic window number 2
Modified Image without preserving Background
USA

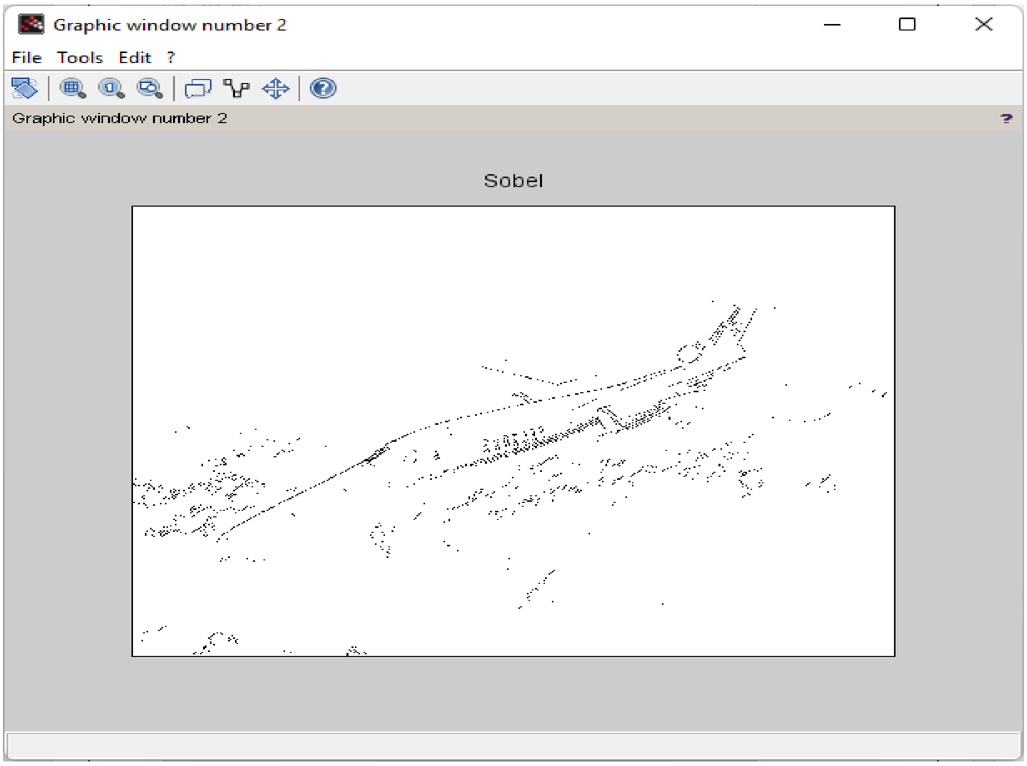
Practical 08

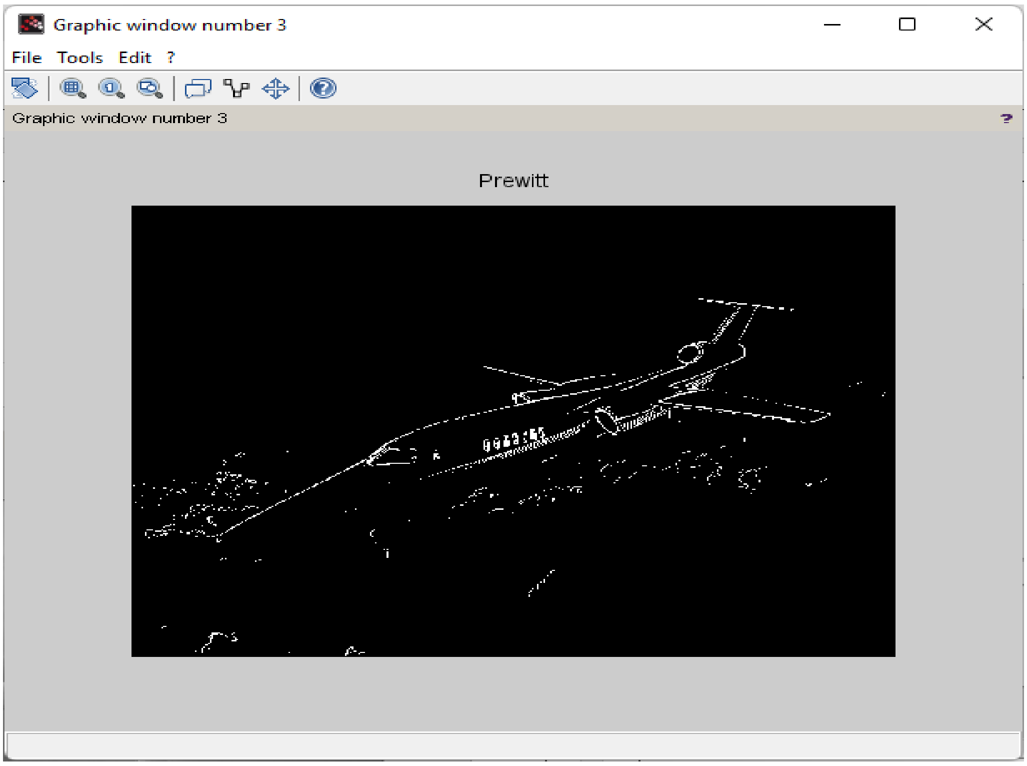
Aim: Image Segmentation.

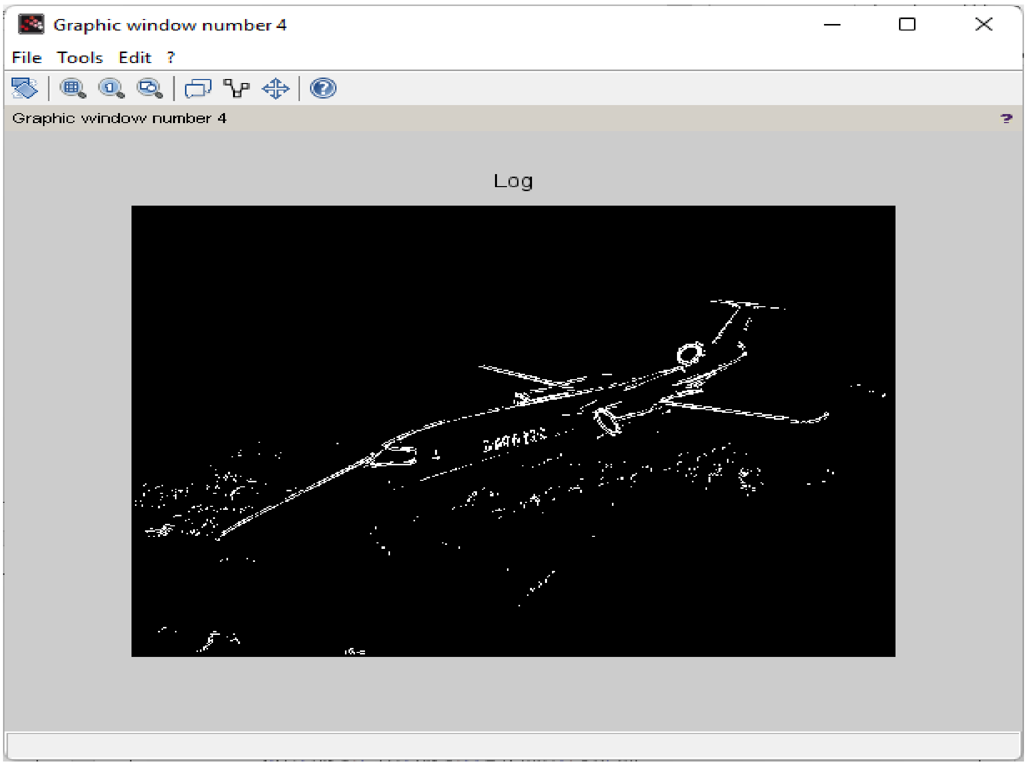
Code and Output:

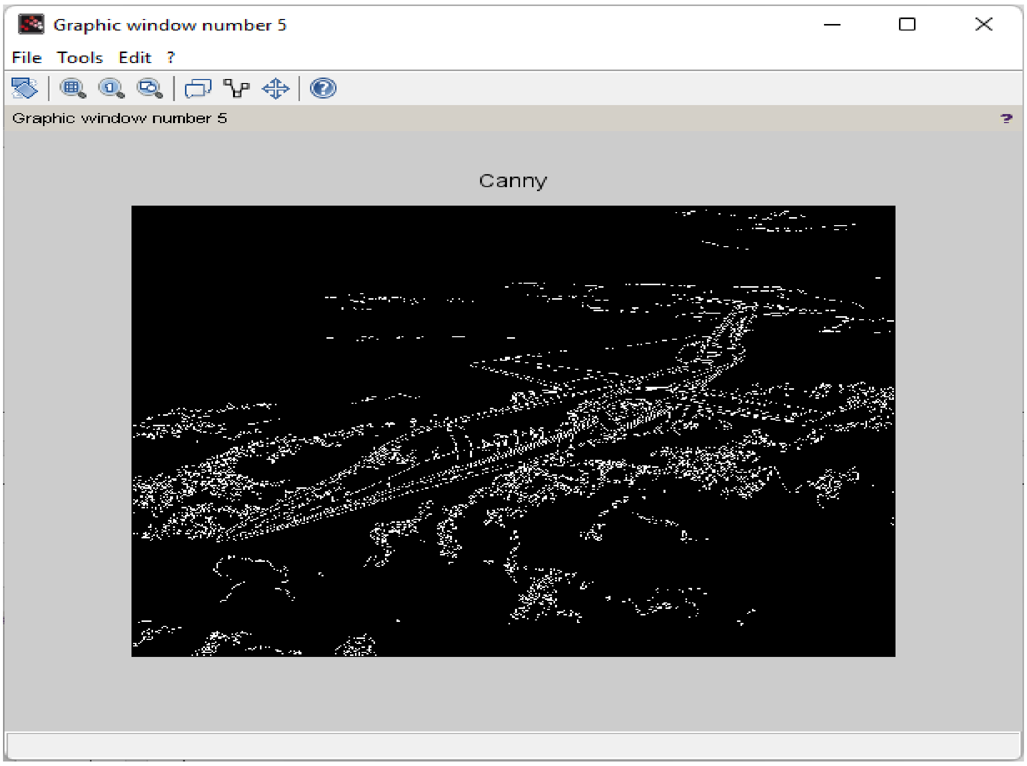
```
clc;
clear all;
close;

a = imread("p8.jpg");
a = rgb2gray(a);
c = edge(a, 'sobel', 0.5);
d = edge(a, 'prewitt');
e = edge(a, 'log');
f = edge (a, 'canny');

figure(1)
imshow(a)
title ('Original Image')

figure(2)
imshow(c)
title ('Sobel')

figure(3)
imshow(d)
title('Prewitt')

figure(4)
imshow(e)
title ('Log')

figure(5)
imshow(f)
title ('Canny')
```


Graphic window number 1
File Tools Edit ?
Graphic window number 1
Original Image


Graphic window number 2
File Tools Edit ?
Graphic window number 2
Sobel


Graphic window number 3
File Tools Edit ?
Graphic window number 3
Prewitt


Graphic window number 4
File Tools Edit ?
Graphic window number 4
Log


Graphic window number 5
File Tools Edit ?
Graphic window number 5
Canny

Practical 09

Aim: Image Compression.

Code and Output:

A. BTC (Block Truncation Coding)

```
clc;
clear all;
close;

x = [65 ,75 ,80 ,70; 72 ,75 ,82 ,68; 84 ,72 ,62 ,65;66 ,68 ,72 ,80];
disp('Original Block is x =', x)
[m1 n1]= size (x);
blk = input('Enter the block size: ');
for i = 1 : blk : m1
   for j = 1 : blk : n1
      y = x(i:i+( blk -1) ,j:j+( blk -1));

      //Step 1: Computation of mean value, sd
      m = mean ( mean (y));
      disp('mean value is m =', m)
      sig = std2 (y);
      disp('Standard deviation of the block is=', sig)

      //Step 2: Computation of binary allocation matrix
      b = y > m ; // the binary block
      disp('Binary allocation matrix is B=', b)
      K = sum (sum (b));
      disp('Number of one s =', K)

      //Step 3: Computation of a and b values
      if (K ~= blk ^2 ) & ( K ~= 0)
        ml = m- sig * sqrt (K/(( blk ^2) -K));
        disp('The value of a =', ml)
        mu = m + sig * sqrt ((( blk ^2) -K)/K);
        disp('The value of b =', mu)
        x(i:i+( blk -1) , j:j+( blk -1) ) = b*mu+(1 - b)*ml;
      end
    end
end
//Step 4: Reconstructed block
```

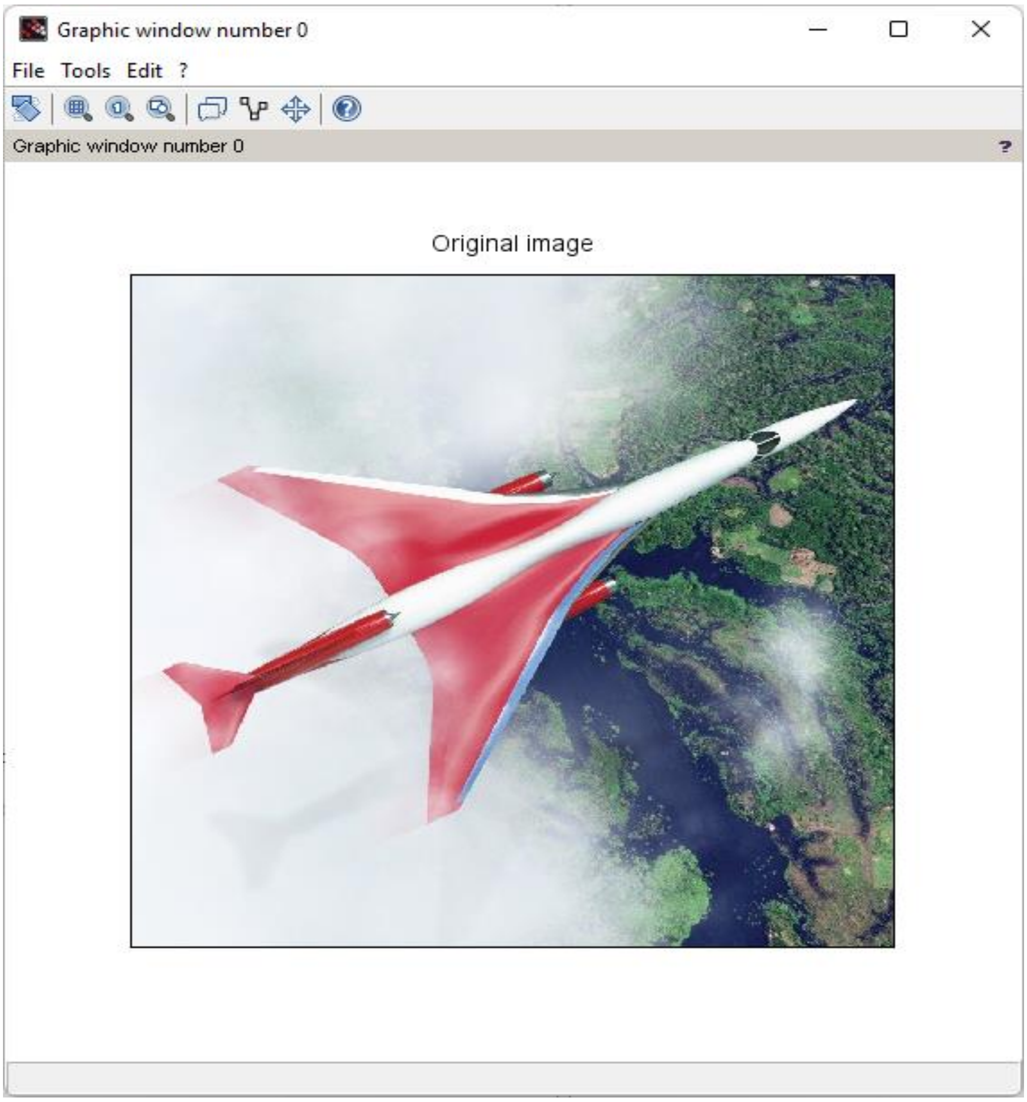
```
disp('Reconstructed Block is x =', round(x))
```

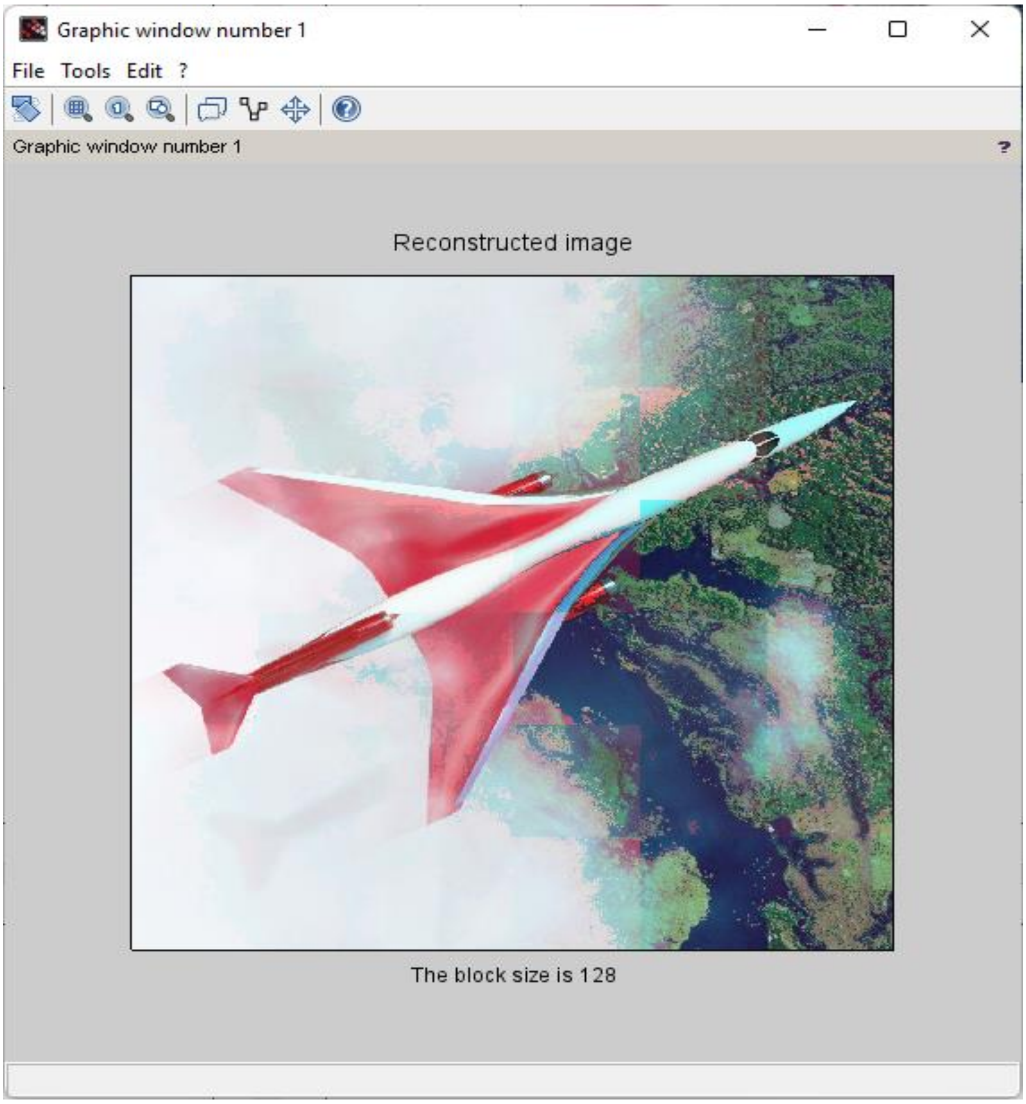
```
Scilab 6.1.1 Console

  "Original Block is x ="

   65.   75.   80.   70.
   72.   75.   82.   68.
   84.   72.   62.   65.
   66.   68.   72.   80.
Enter the block size: 4


  "mean value is m ="

   72.25

  "Standard deviation of the block is="

   6.6282225

  "Binary allocation matrix is B="

  F T T F
  F T T F
  T F F F
  F F F T

  "Number of one s ="

   6.

  "The value of a ="

   67.115801

  "The value of b ="

   80.806998

  "Reconstructed Block is x ="

   67.   81.   81.   67.
   67.   81.   81.   67.
   81.   67.   67.   67.
   67.   67.   67.   81.

-->
```

B. Image Compression using BTC

```
clc;
clear all;
close;

x = imread('p9.jpg');
//x=rgb2gray(x);
x=double(x);
x1=x;
```

```
[m1 n1]= size (x);
blk = input('Enter the block size: ');
for i = 1 : blk : m1
   for j = 1 : blk : n1
      y = x(i:i+( blk -1) ,j:j+( blk -1));

      //Step 1: Computation of mean value, sd
      m = mean ( mean (y));
      disp (m, 'mean value is m =' )
      sig = std2 (y);
      disp('Standard deviation of the block is=', sig)

      //Step 2: Computation of binary allocation matrix
      b = y > m ; // the binary block
      disp('Binary allocation matrix is B=', b)
      K = sum (sum (b));
      disp('Number of one s =', K)

      //Step 3: Computation of a and b values
      if (K ~= blk ^2 ) & ( K ~= 0)
        ml = m- sig * sqrt (K/(( blk ^2) -K));
        disp('The value of a =', ml)
        mu = m + sig * sqrt ((( blk ^2) -K)/K);
        disp('The value of b =', mu)
        x(i:i+( blk -1) , j:j+( blk -1) ) = b*mu+(1 - b)*ml;
      end
    end
end
//Step 4: Reconstructed block
imshow(uint8(x1)), title('Original image')
figure, imshow(uint8(x)), title('Reconstructed image')
xlabel(sprintf('The block size is %g', blk))
```


Graphic window number 0
File Tools Edit ?
Graphic window number 0
Original image


Graphic window number 1
File Tools Edit ?
Graphic window number 1
Reconstructed image
The block size is 128

Practical 10

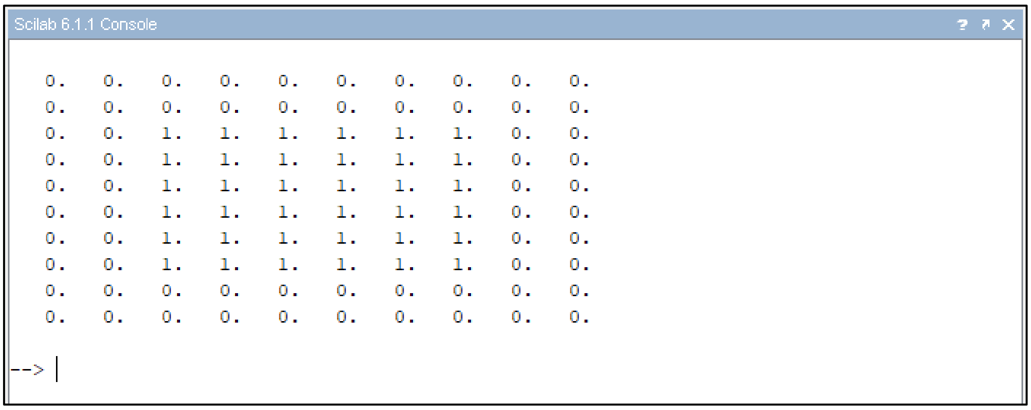
Aim: Binary Image Processing and Color Image processing.

Code and Output:

A. Binary Image Processing

1. Imdilate()

```
clc;
clear all;
close;
a = zeros(10,10);
a(4:7,4:7) = 1;
se = imcreatese('rect',3,3);
b = imdilate(a,se);
disp(b);
```



2. Imerode()

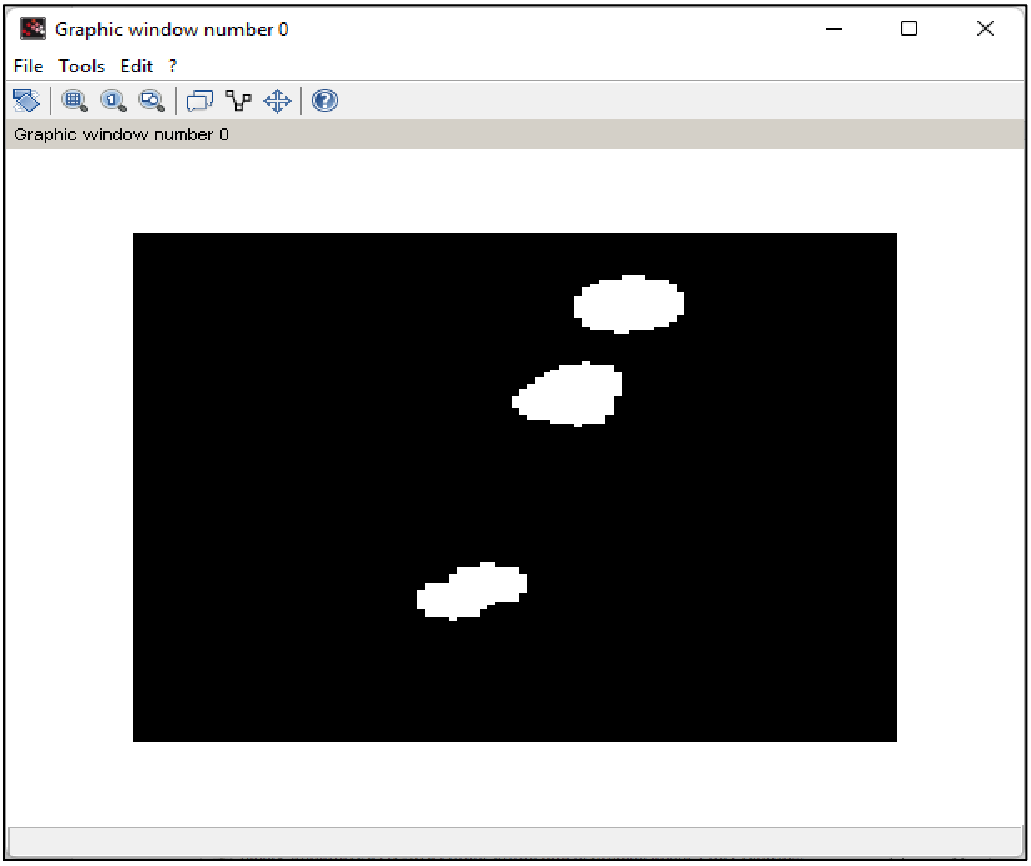
```
clc;
clear all;
close;
a = zeros(10,10);
a(4:7,4:7) = 1;
se = imcreatese('rect',3,3);
b = imerode(a,se);
disp(b);
```

Scilab 6.1.1 Console

```
  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
  0.   0.   0.   0.   1.   1.   0.   0.   0.   0.
  0.   0.   0.   0.   1.   1.   0.   0.   0.   0.
  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.

-->
```

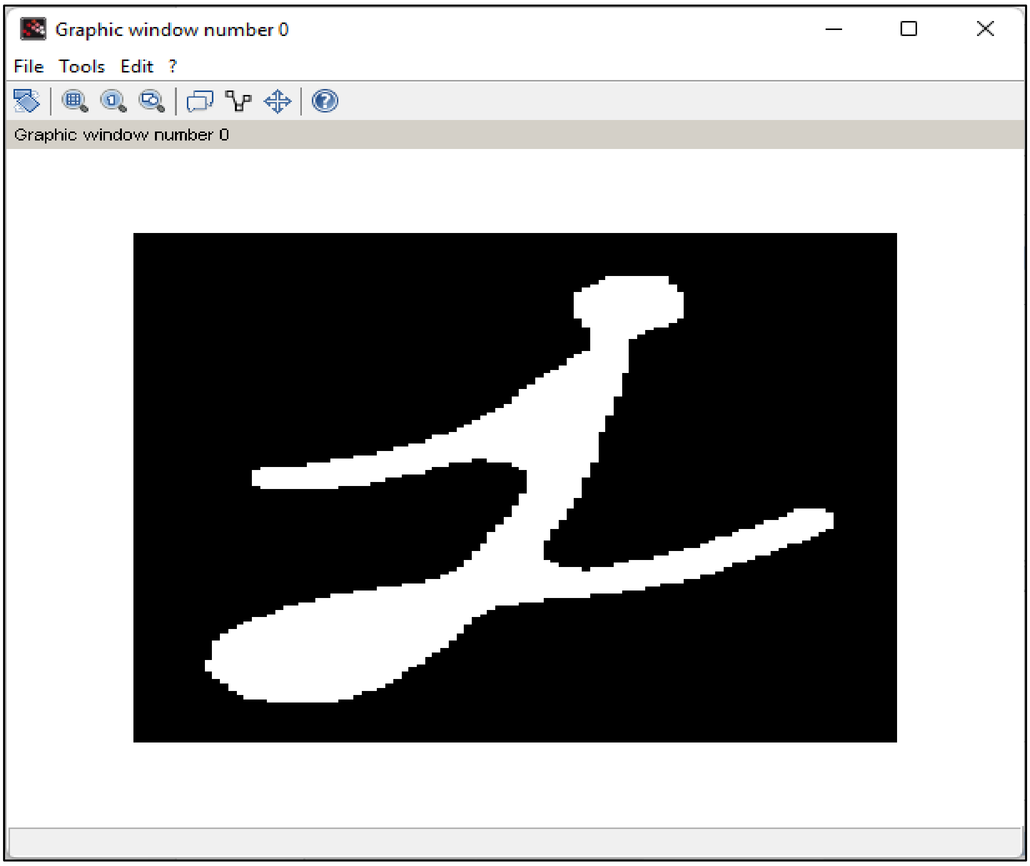
3. Imopen()

```
clc;
clear all;
close;
S = imread(fullpath(getIPCVpath() + "/images/morpex.png"));
se = imcreatese('ellipse',9,9);
S2 = imopen(S,se);
imshow(S2);
```



4. **Imclose()**

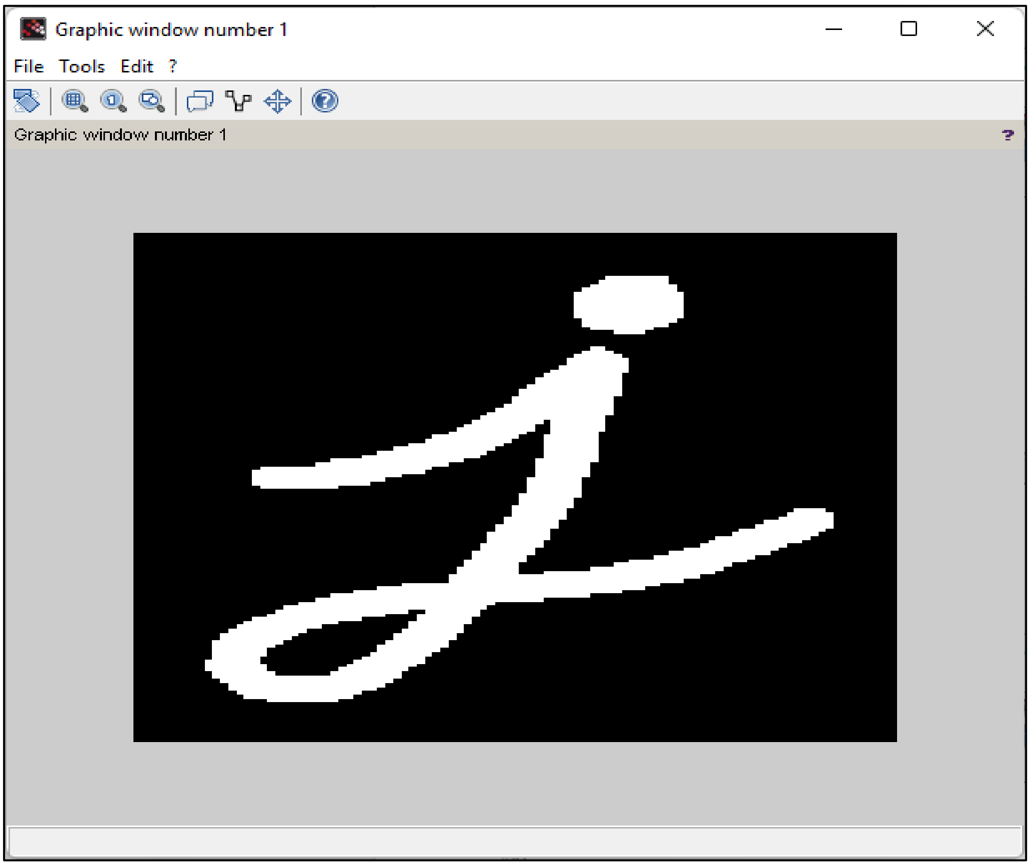
```
clc;
clear all;
close;
S = imread(fullpath(getIPCVpath() + "/images/morpex.png"));
se = imcreatese('ellipse',11,11);
S2 = imclose(S,se);
imshow(S2);
```

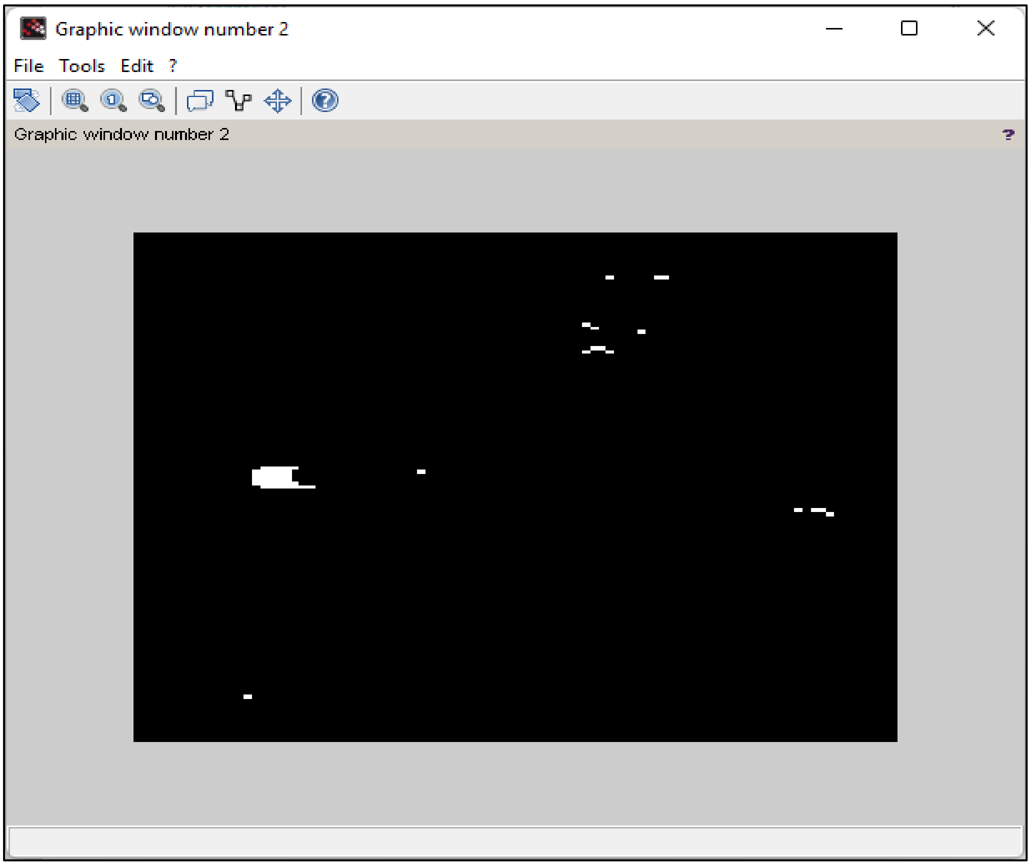


5. **Imtophat()**

```
clc;
clear all;
close;
S = imread(fullpath(getIPCVpath() + "/images/morpex.png"));
se = imcreatese('ellipse', 7,7);
S2 = imtophat(S,se);
figure(1)
imshow(S);
figure(2)
```

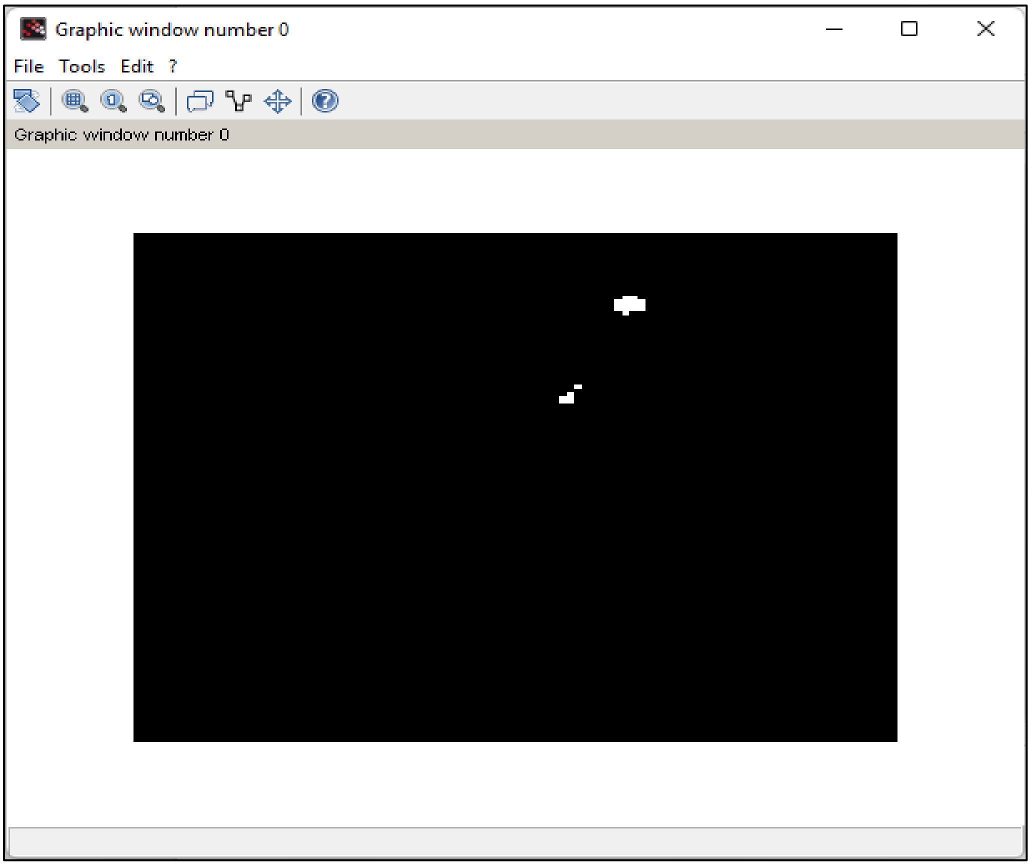
imshow(S2);


Graphic window number 1
File Tools Edit ?
Graphic window number 1


Graphic window number 2
File Tools Edit ?
Graphic window number 2

6. Imhitmiss()

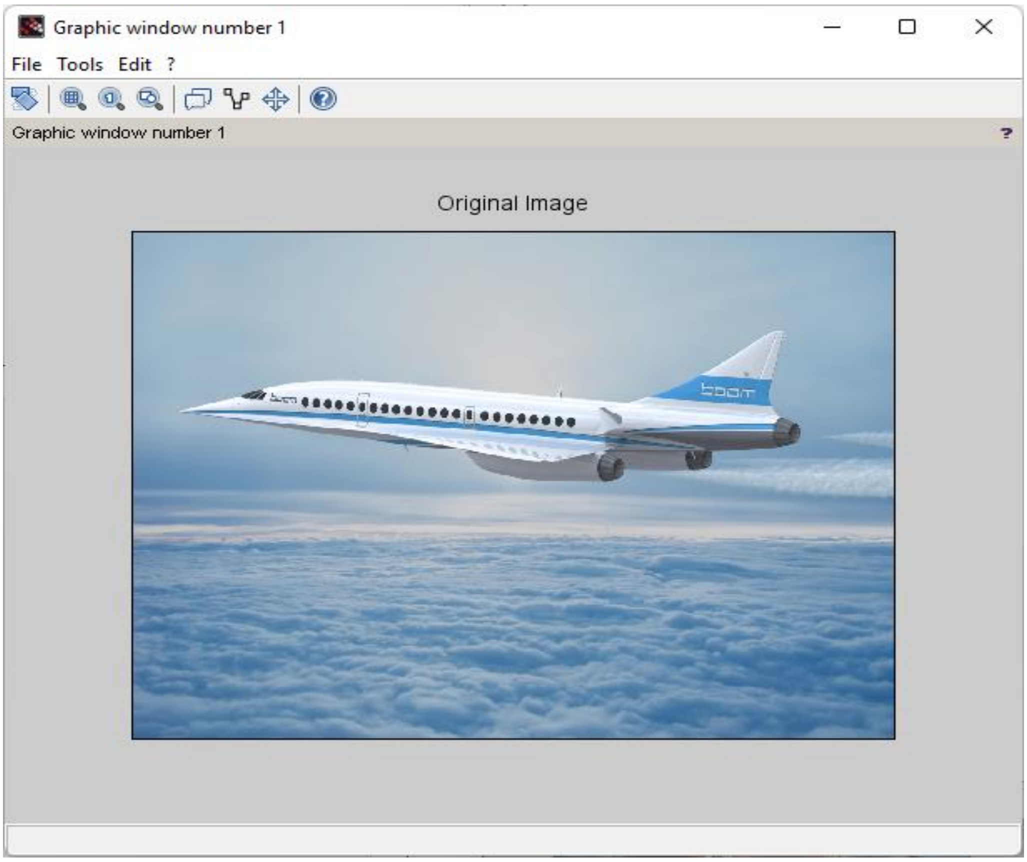
```
clc;
clear all;
close;
S = imread(fullpath(getIPCVpath() + "/images/morpex.png"));
se = imcreatese('ellipse',11,11);
S2 = imhitmiss(S,se);
imshow(S2);
```

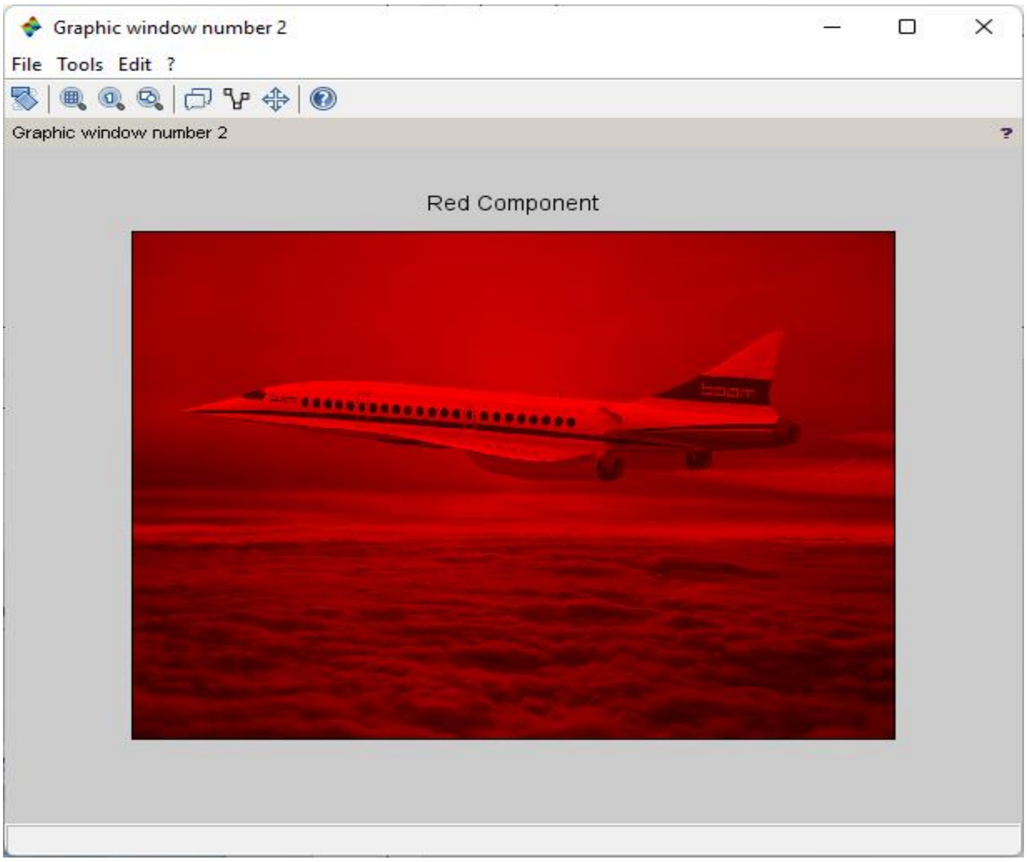


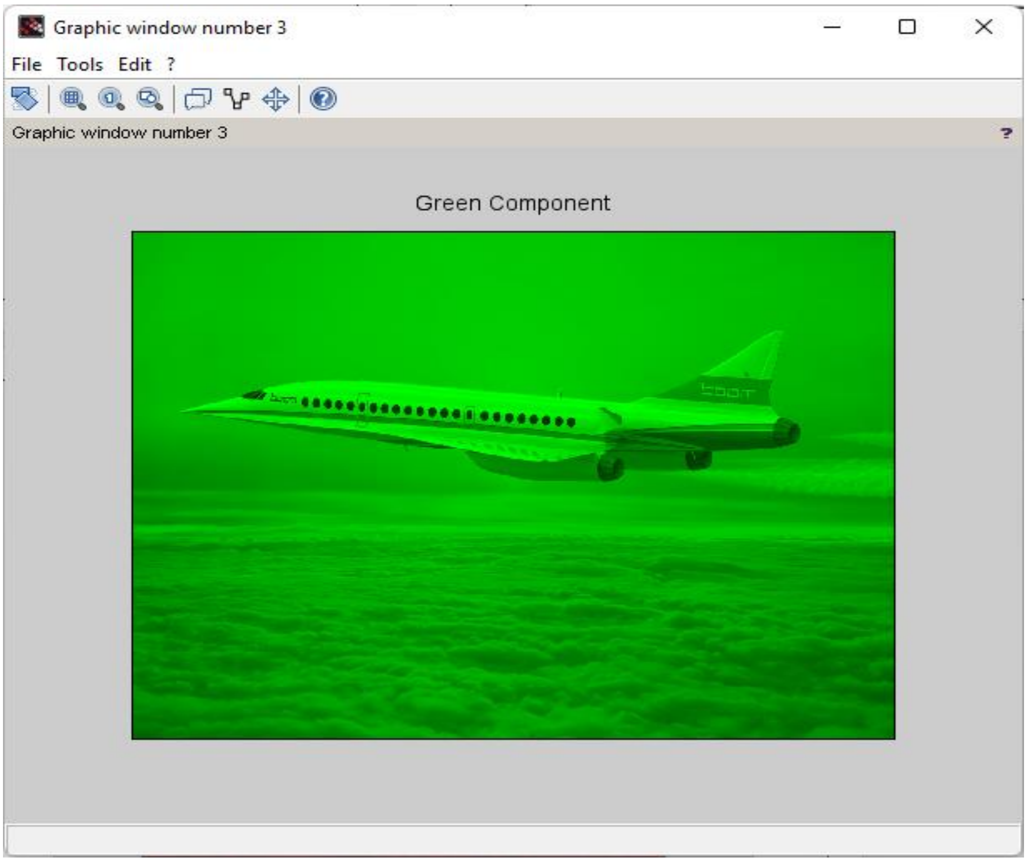
B. Color Image Processing

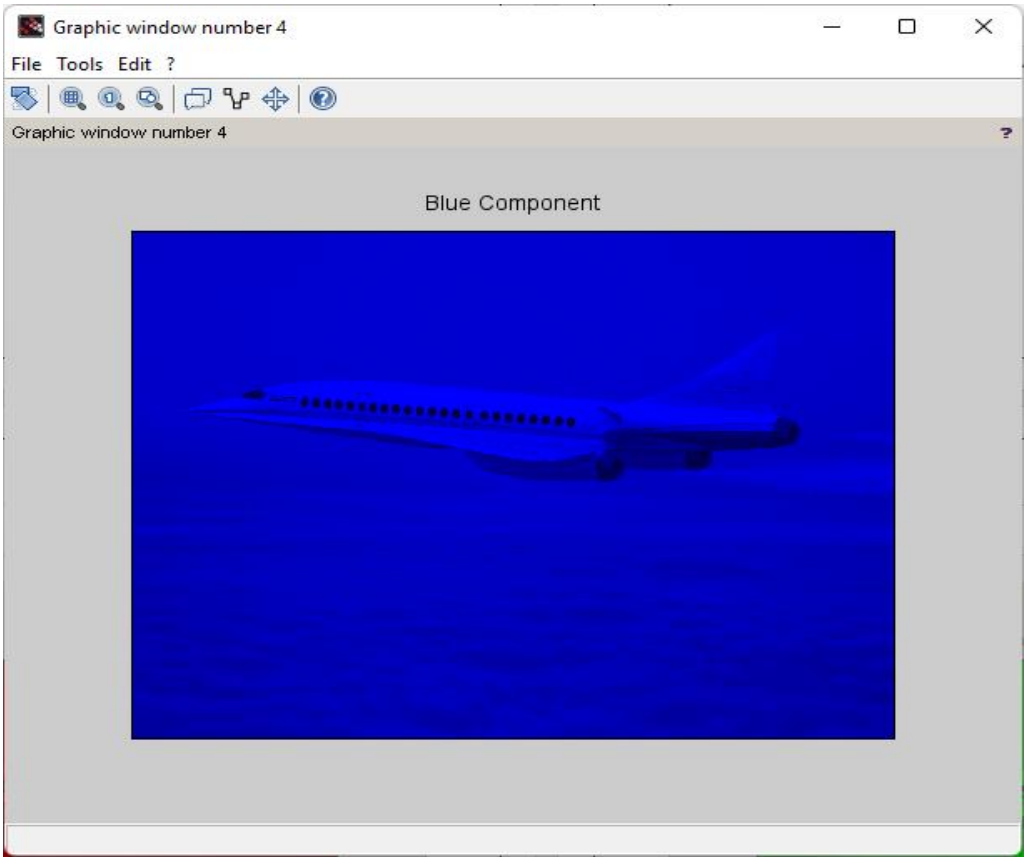
```
clc;
clear all;
close;
RGB = imread('p10.jpg');
R = RGB;
G = RGB;
B = RGB;
R(: ,: ,2) =0;
R(: ,: ,3) =0;
```

```
G(: ,: ,1) =0;
G(: ,: ,3) =0;
B(: ,: ,1) =0;
B(: ,: ,2) =0;
figure (1)
imshow(RGB)
title('Original Image');
figure (2)
imshow(R)
title('Red Component');
figure (3)
imshow(G)
title('Green Component');
figure (4)
imshow(B)
title('Blue Component');
```




Graphic window number 2
File Tools Edit ?
Graphic window number 2
Red Component


Graphic window number 3
File Tools Edit ?
Graphic window number 3
Green Component


Graphic window number 4
File Tools Edit ?
Graphic window number 4
Blue Component