EXPERIMENT NO. 04

AIM: LINEAR ALZEBRA WITH VARIOUS APPLICATIONS.

NAME: Soniya Eknath Girhe.

SEC_Batch_RollNo.: A7_B3_59

DATE: 11/02/26

In [1]:
```
def matrix_transformation(A,L):
    n=matrix(L).nrows()
    L2=[[0,0] for i in range (n)]
    for i in range (n):
        L2[i]=list(A*vector(L[i]))
    return L2
print ("The matrix_transformation function is activated.")
```

```
The matrix_transformation function is activated.
```

In [2]:
```
A=matrix(2,2,[3,0,0,3])
B=matrix(2,2,[1,1,0,1])
C=matrix(2,2,[cos(pi/2),-sin(pi/2),sin(pi/2),cos(pi/2)])
D=matrix(2,2,[-1,0,0,-1])
show(A,B,C,D)
```

$$\begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

In [3]:
```
A=matrix(2,2,[3,0,0,3])
A
```

Out[3]:
```
[3 0]
[0 3]
```

In [4]:
```
B=matrix(2,2,[1,1,0,1])
B
```

Out[4]:
```
[1 1]
[0 1]
```

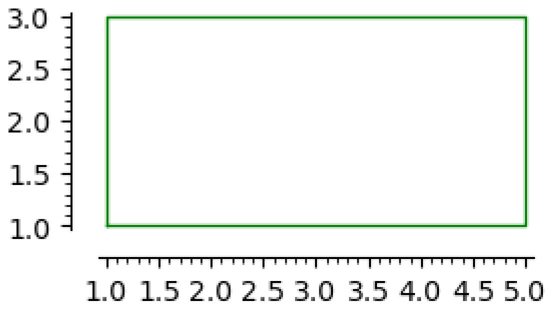
In [5]:
```
C=matrix(2,2,[cos(pi/2),-sin(pi/2),sin(pi/2),cos(pi/2)])
C
```

Out[5]:
```
[ 0 -1]
[ 1  0]
```

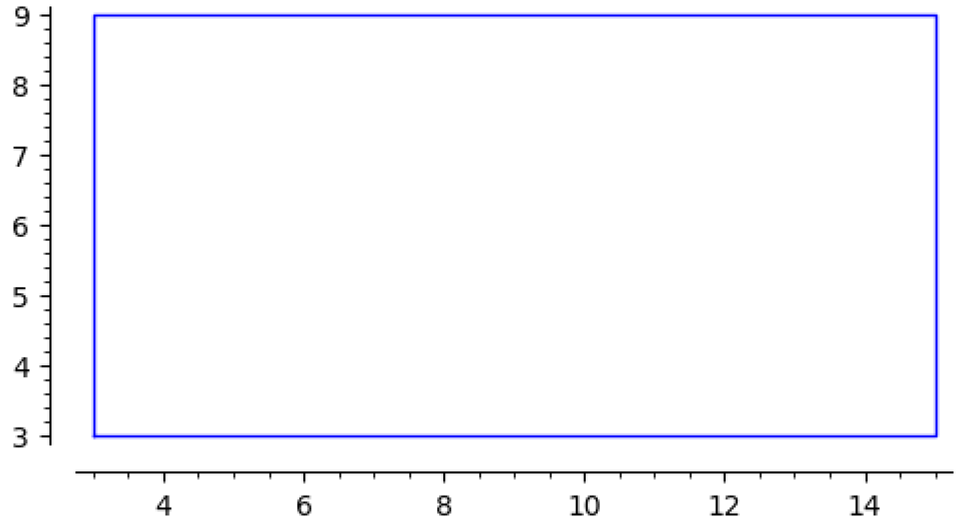
In [6]:
```
D=matrix(2,2,[-1,0,0,-1])
D
```

Out[6]:
```
[-1  0]
[ 0 -1]
```

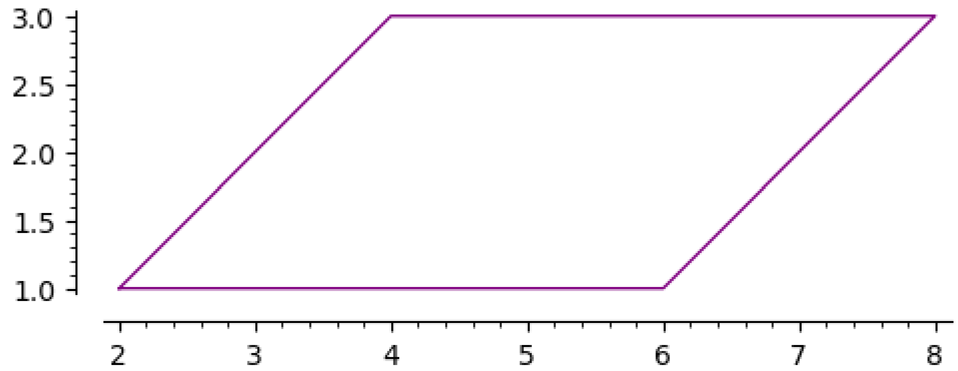
In [7]:
```
L1=list([[1,1],[5,1],[5,3],[1,3],[1,1]])
SL1=line(L1,color="green")
SL1.show(aspect_ratio=1,figsize=3)
```



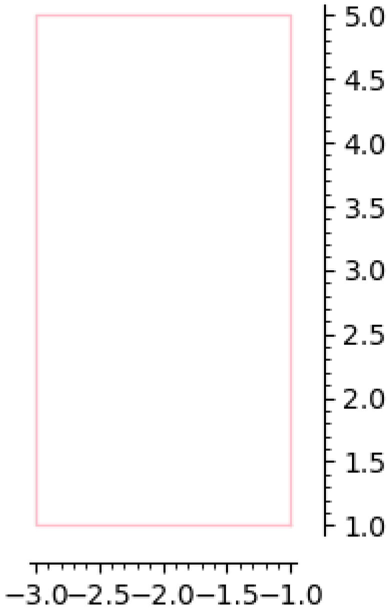
In [8]:
```
L2=matrix_transformation(A,L1)
SL2=line(L2,color="blue")
SL2.show(aspect_ratio=1,figsize=5)
```



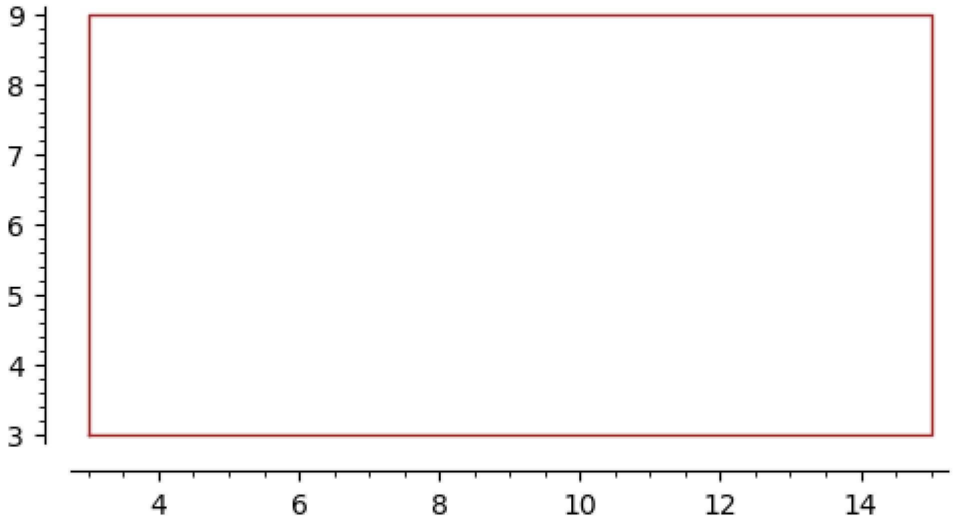
In [9]:
```
L3=matrix_transformation(B,L1)
SL3=line(L3,color="purple")
SL3.show(aspect_ratio=1,figsize=5)
```



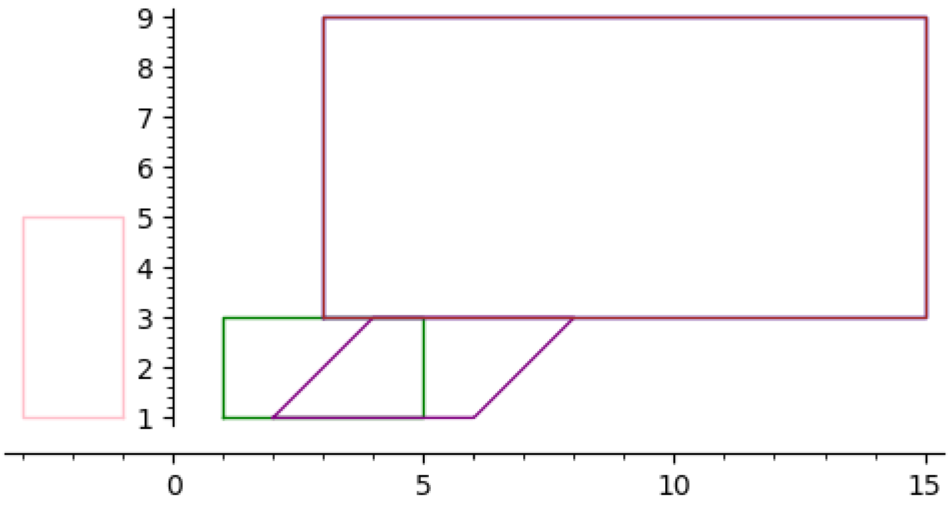
In [10]:
```
L4=matrix_transformation(C,L1)
SL4=line(L4,color="pink")
SL4.show(aspect_ratio=1,figsize=5)
```



In [11]:
```
L5=matrix_transformation(D,L1)
SL5=line(L2,color="brown")
SL5.show(aspect_ratio=1,figsize=5)
```



In [12]:
```
(SL1+SL2+SL3+SL4+SL5).show(aspect_ratio=1,figsize=5)
```



In [13]:
```
A=matrix(CDF,3,3,[1,2,3,4,5,6,7,8,9])
show(A)
```

$$\begin{pmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 7.0 & 8.0 & 9.0 \end{pmatrix}$$

In [14]:
```
U,S,V=A.SVD()
```

In [15]:
```
show(U)
```

$$\begin{pmatrix} -0.21483723836839674 & 0.8872306883463708 & 0.4082482904638625 \\ -0.5205873894647373 & 0.24964395298829734 & -0.8164965809277261 \\ -0.8263375405610782 & -0.3879427823697743 & 0.4082482904638633 \end{pmatrix}$$

In [16]:
```
show(S)
```

$$\begin{pmatrix} 16.84810335261421 & 0.0 & 0.0 \\ 0.0 & 1.06836951455471 & 0.0 \\ 0.0 & 0.0 & 4.4184247511933675 \times 10^{-16} \end{pmatrix}$$

In [17]:
```
U*S*(V.transpose())
```

Out[17]:
```
[1.0000000000000022  2.000000000000005   3.000000000000007]
[ 4.000000000000001  5.000000000000003 6.0000000000000036]
[ 7.000000000000002  8.000000000000007   9.000000000000007]
```

In [18]:
```
show(V)
```

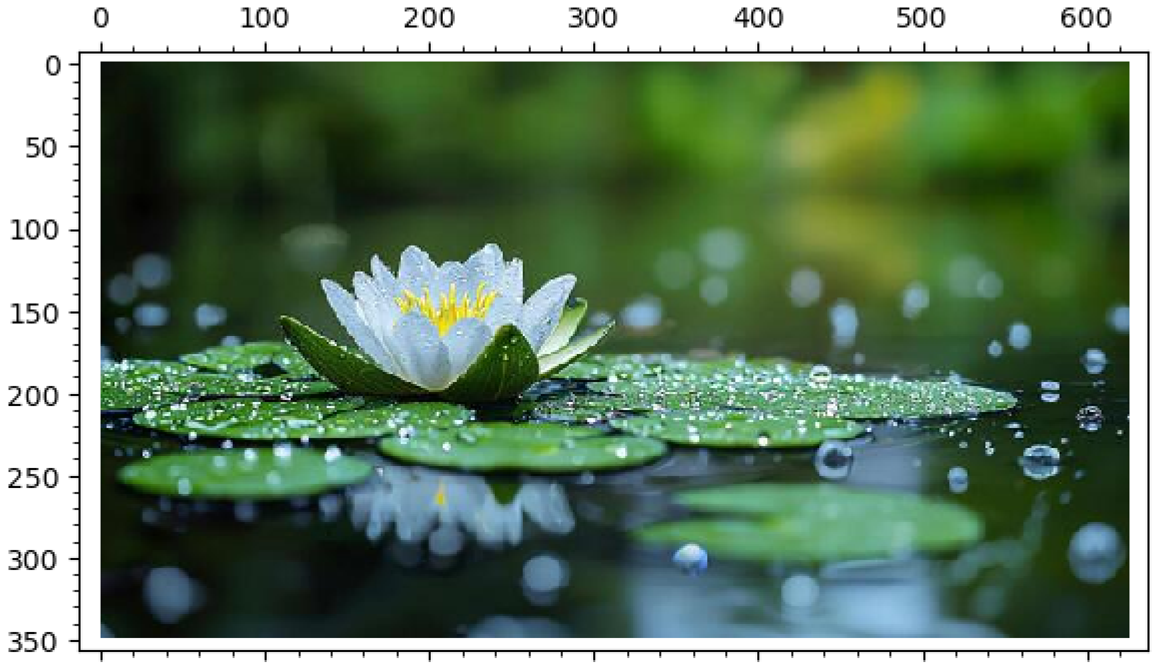
$$\begin{pmatrix} -0.4796711778777147 & -0.7766909903215595 & -0.4082482904638631 \\ -0.5723677939720624 & -0.07568647010455853 & 0.8164965809277263 \\ -0.6650644100663531 & 0.6253180501124426 & -0.4082482904638631 \end{pmatrix}$$

dimension Reducting using SVD

In [19]:
```
from matplotlib.pyplot import imread
import pylab
import numpy as np
img=pylab.imread('img1.png')
```

In [20]:
```
matrix_plot(img)
```

Out[20]:



In [29]:
```
img.shape
```

Out[29]: (350, 625, 3)

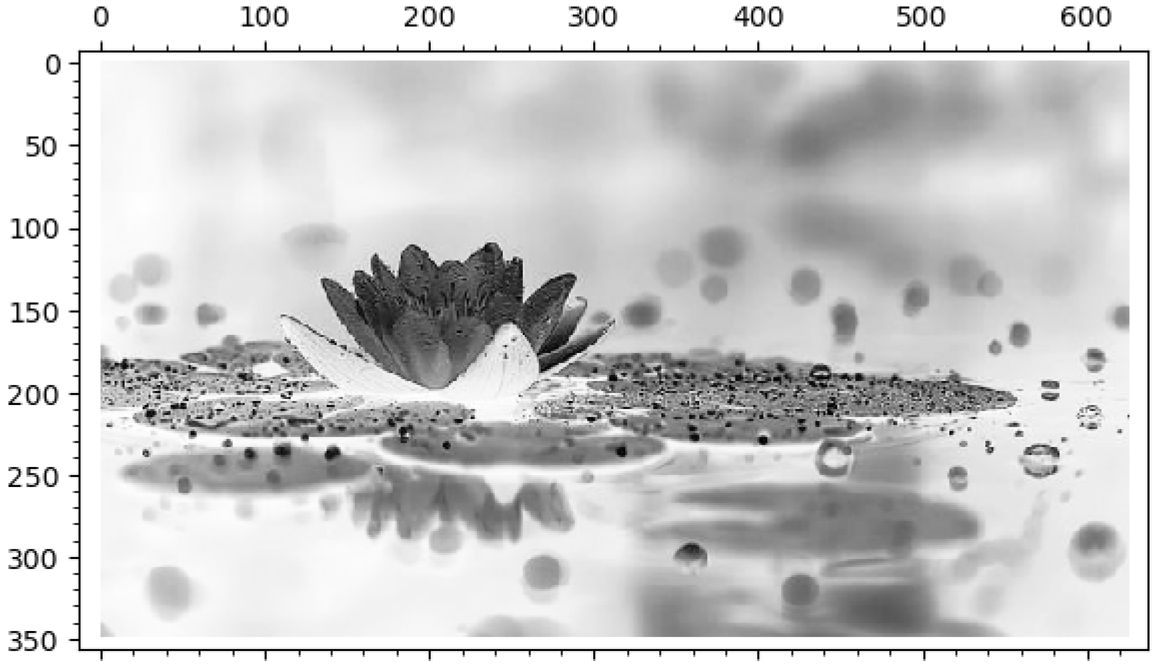
In [22]:
```
show(img)
```

```
[[[0.04313726 0.05882353 0.05490196]
  [0.04313726 0.05882353 0.05490196]
  [0.04313726 0.05882353 0.05490196]
  ...
  [0.15686275 0.2901961  0.05490196]
  [0.15686275 0.2901961  0.05098039]
  [0.15686275 0.2901961  0.05098039]]

 [[0.04313726 0.05882353 0.05490196]
  [0.04313726 0.05882353 0.05490196]
  [0.04313726 0.05882353 0.05490196]
  ...
  [0.16078432 0.29411766 0.05490196]
  [0.16078432 0.29411766 0.05490196]
  [0.16078432 0.29803923 0.04705882]]

 [[0.04313726 0.0627451  0.04705882]
  [0.04313726 0.0627451  0.04705882]
  [0.04313726 0.0627451  0.04705882]
  ...
  [0.16862746 0.30588236 0.05490196]
  [0.16862746 0.30588236 0.04705882]
  [0.16862746 0.30588236 0.04705882]]

 ...

 [[0.22352941 0.35686275 0.4627451 ]
  [0.22352941 0.35686275 0.45490196]
  [0.23137255 0.35686275 0.44705883]
  ...
  [0.0627451  0.13333334 0.08627451]
```

```
 [0.0627451  0.13333334 0.08627451]
 [0.0627451  0.13333334 0.07843138]]

[[0.23137255 0.36862746 0.4862745 ]
 [0.22352941 0.36078432 0.47058824]
 [0.23137255 0.3529412  0.4627451 ]
 ...
 [0.0627451  0.13333334 0.08627451]
 [0.0627451  0.13333334 0.08627451]
 [0.0627451  0.13333334 0.08627451]]

[[0.23137255 0.37254903 0.49803922]
 [0.22745098 0.3647059  0.48235294]
 [0.21960784 0.3529412  0.4627451 ]
 ...
 [0.0627451  0.13333334 0.08627451]
 [0.05882353 0.12941177 0.08235294]
 [0.05882353 0.12941177 0.08235294]]]
```

In [23]:
```
gray=lambda rgb: np.dot(rgb[...,:3],[0.3,0.6,0.1])
G=gray(img)
matrix_plot(G)
```

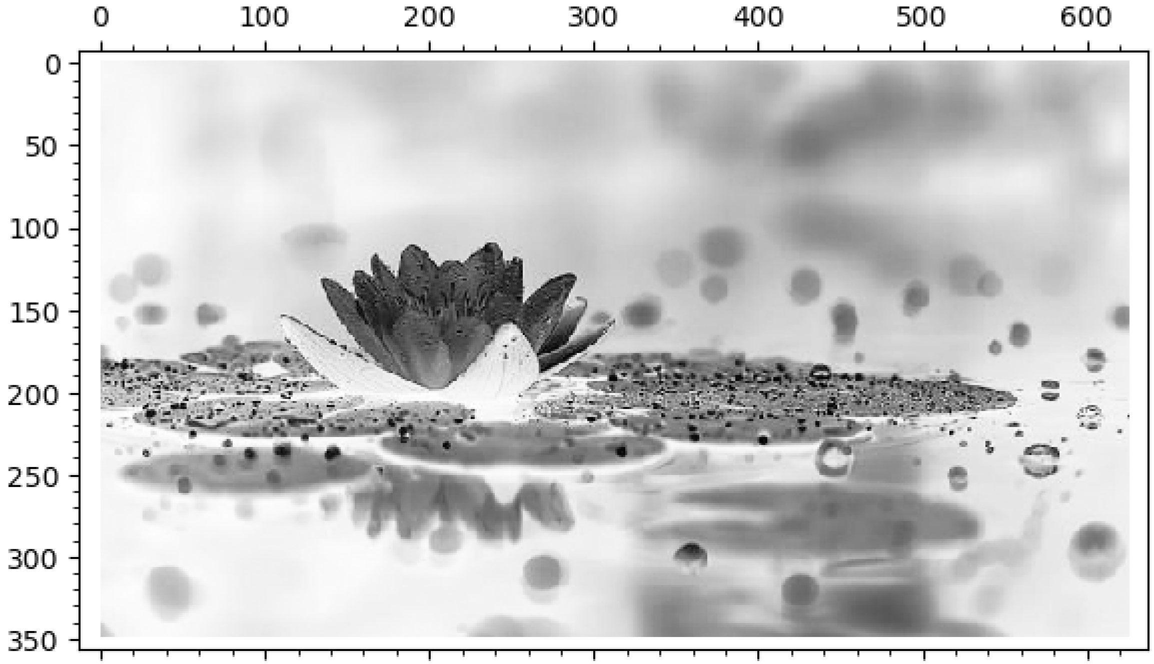
Out[23]:



In [24]:
```
U,S,V=matrix(G).SVD()
```

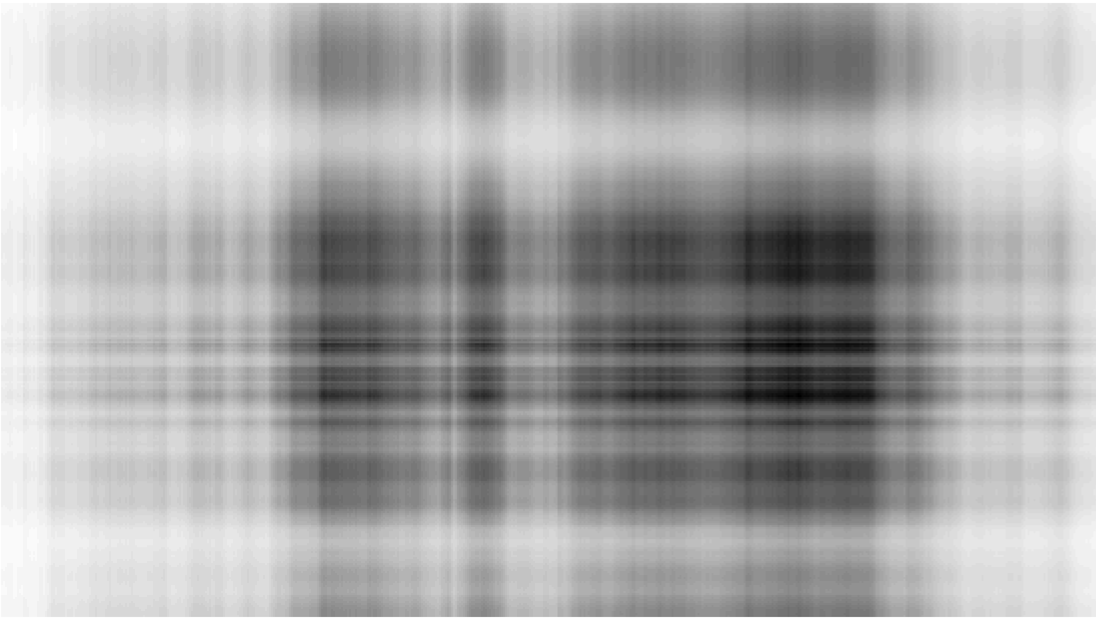
In [25]:
```
n=200
G1=U[:,:n]*S[:n,:n]*V.T[:n,:]
reduced=matrix_plot(G1)
reduced
```

Out[25]:

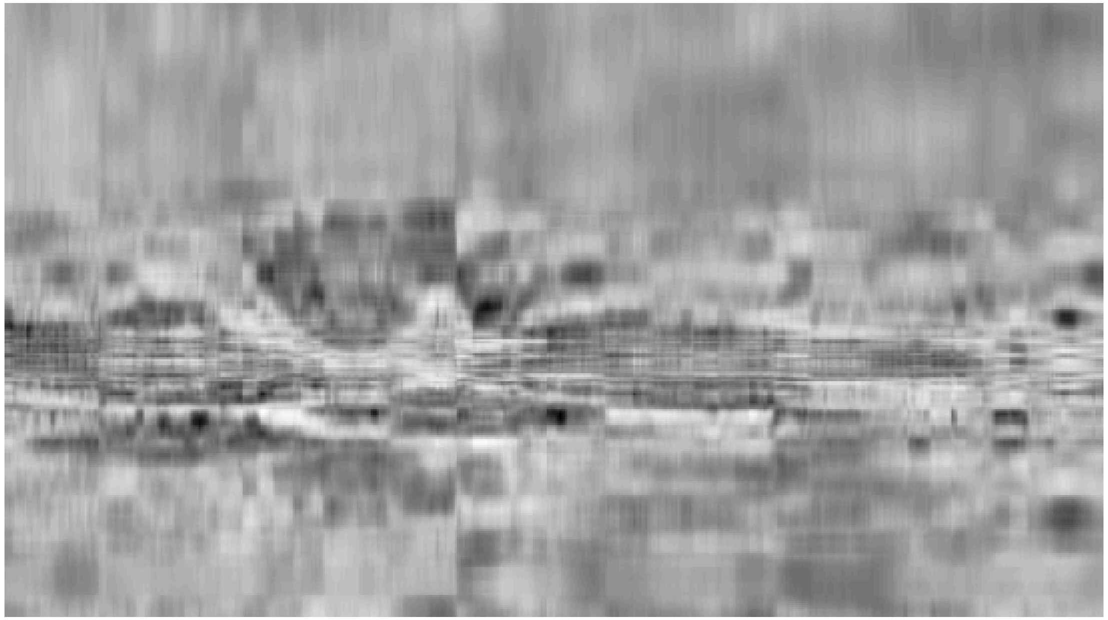


In [26]:
```
appx =[]
for i in range(1,100,10):
    A_approx = U[:,:i]*V.T[:i,:]
    appx_img=matrix_plot(A_approx,title="Using "+str(i)+'Singular Values',
                         frame=False)
    show(appx_img,figsize=6)
```

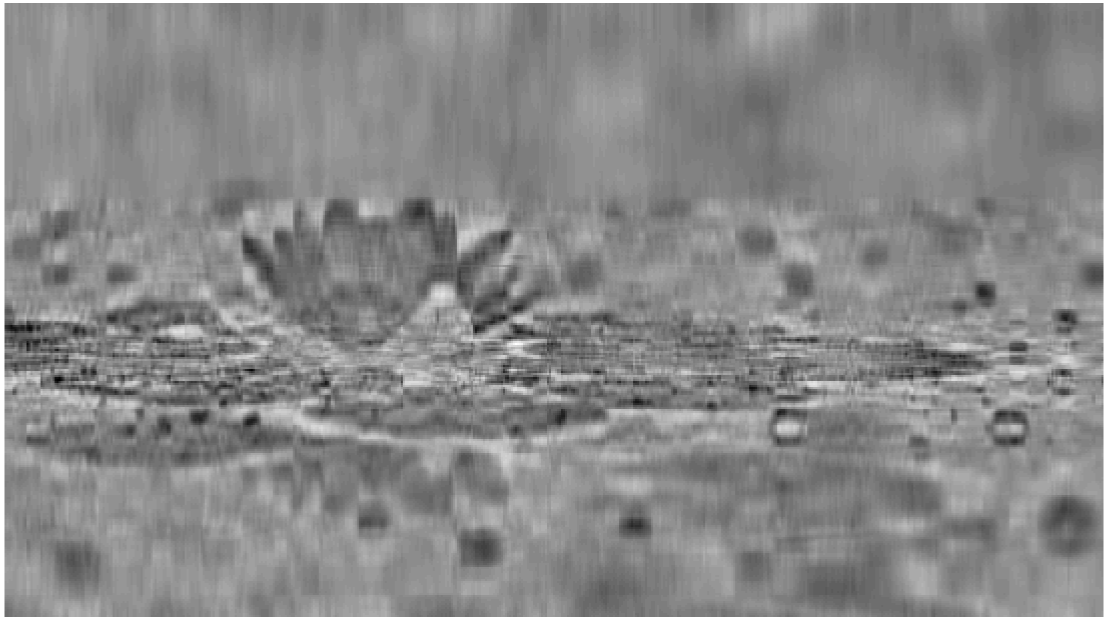
Using 1Singular Values



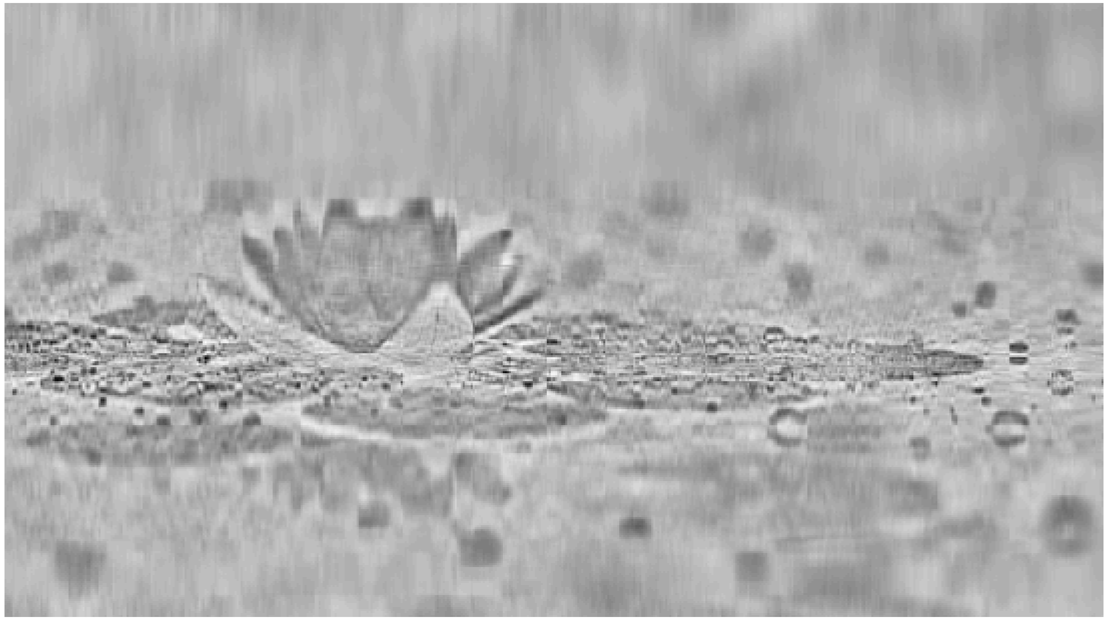
Using 11Singular Values



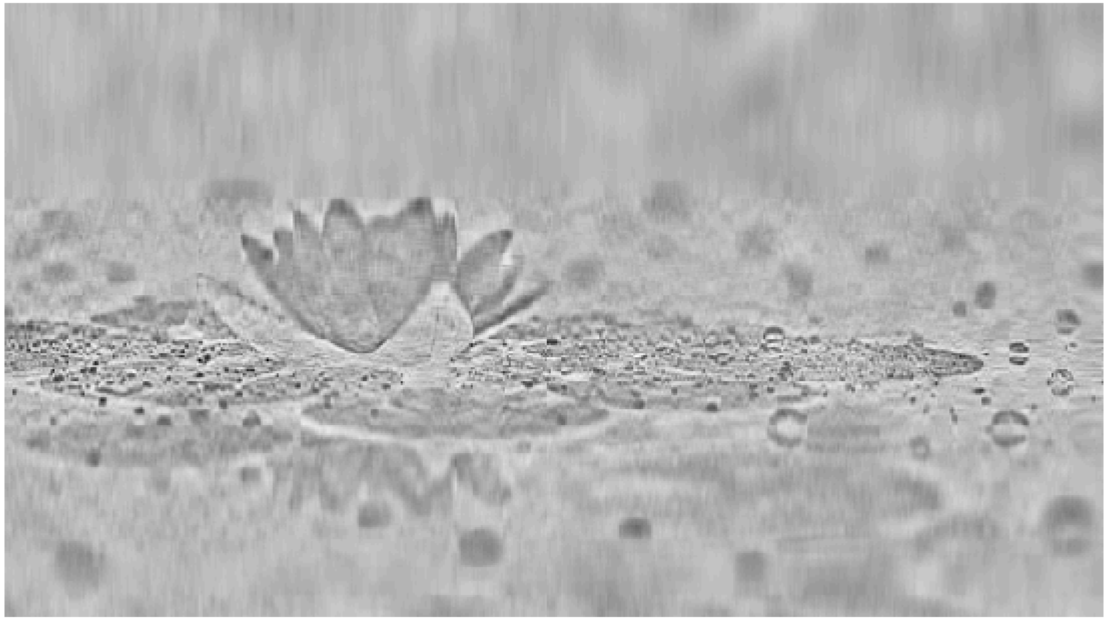
Using 21Singular Values



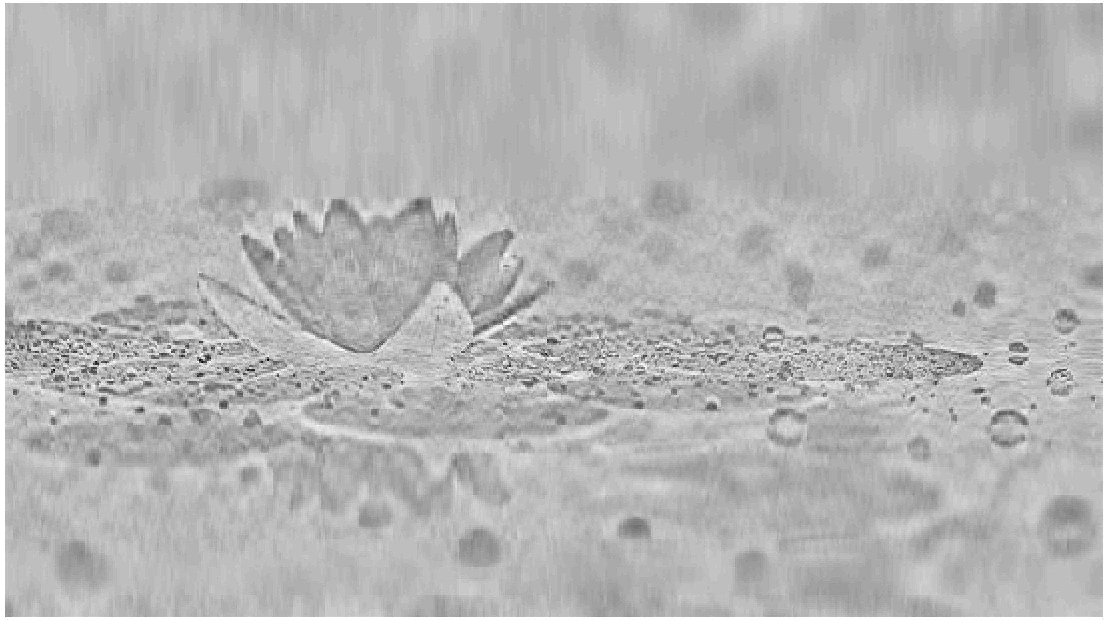
Using 31Singular Values



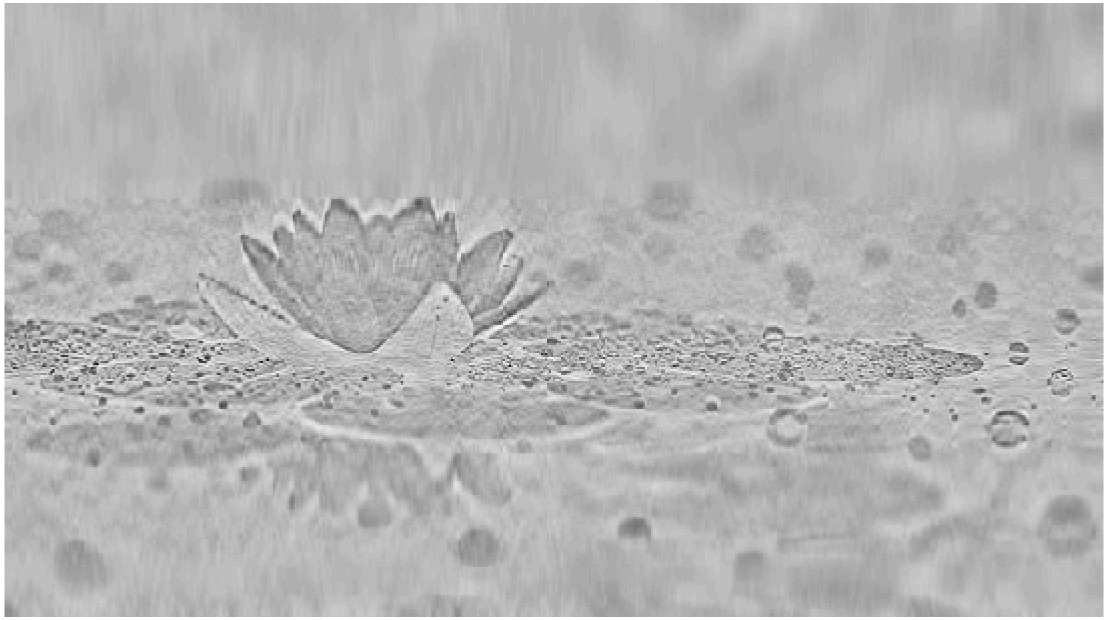
Using 41Singular Values



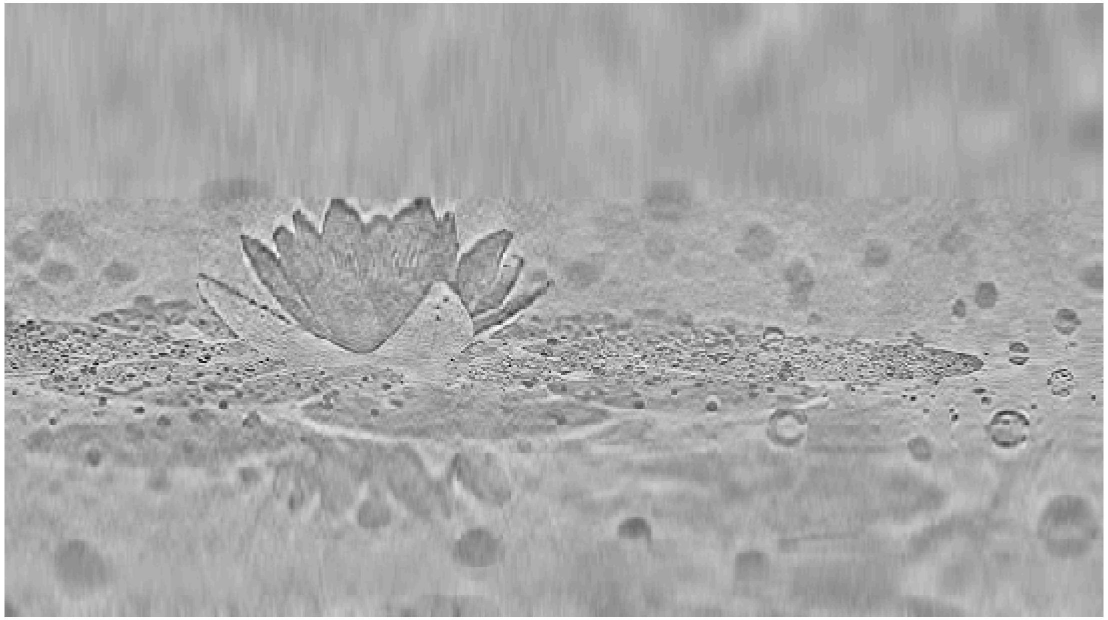
Using 51Singular Values



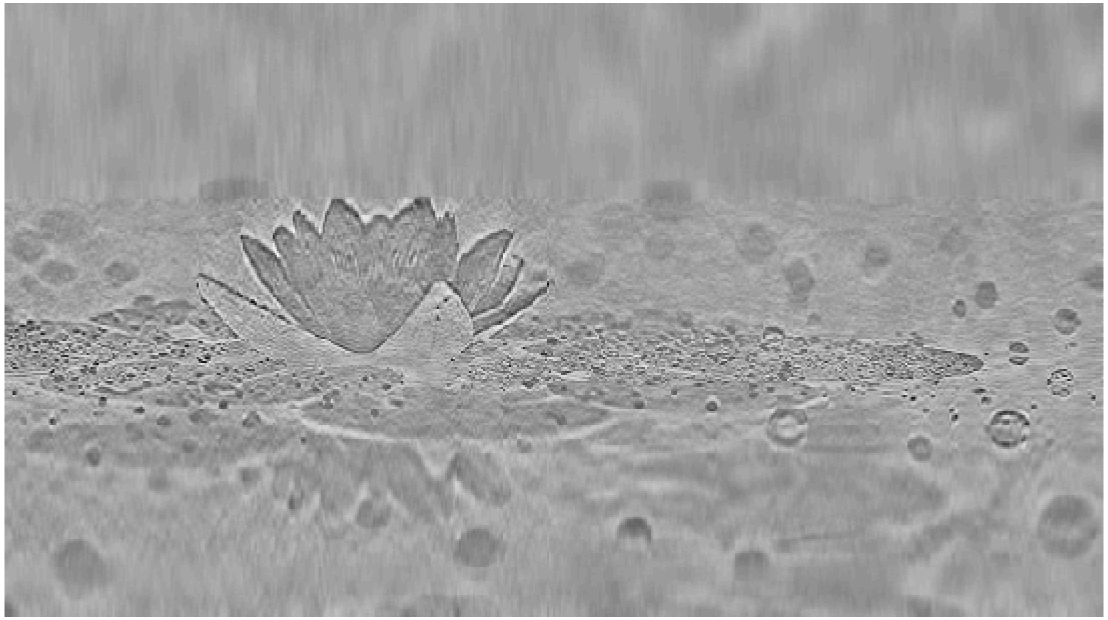
Using 61Singular Values



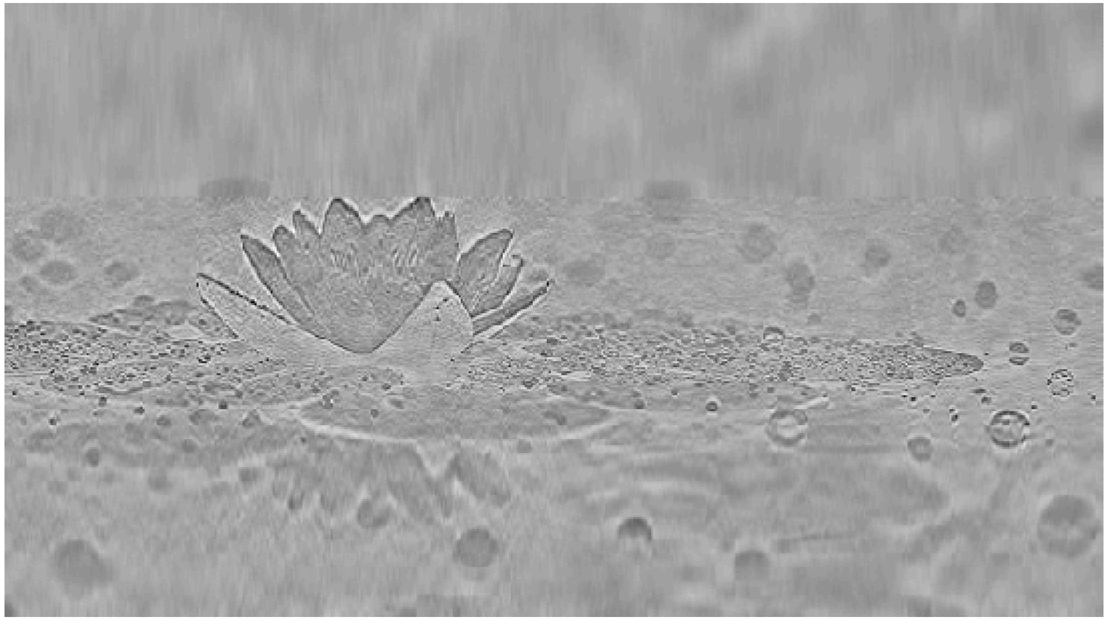
Using 71Singular Values



Using 81Singular Values



Using 91Singular Values



Exercise

1. Construct a shape by joining the points

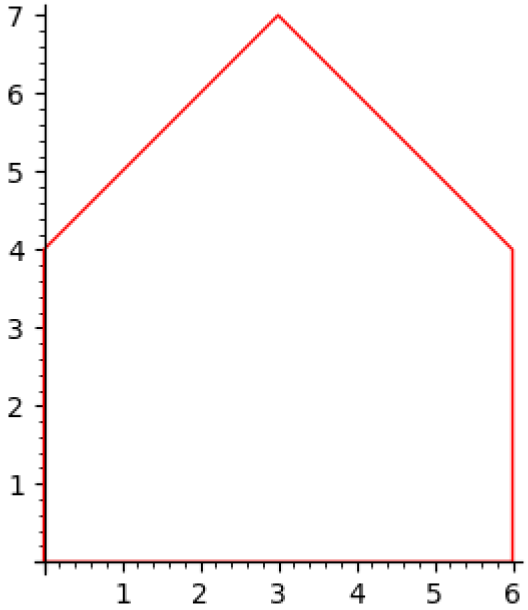
(0,0),(6,0),(6,4),(3,7),(0,4),(0,0). Apply proper matrix transformations for: Expanding to twice the given size. Shearing along the X-axis with a scale of 1. Rotating counter-clockwise by an angle of π/2.

In [35]:
```
A=matrix([[3,0], [0,3]]) # Expanding twice of given image
B=matrix([[1,1], [0,1]]) # shear transformation along the x-axis with scale
C = matrix([[cos(pi/2), -sin(pi/2)], [sin(pi/2), cos(pi/2)]])
D=matrix ([[-1,0],[0,1]])
```

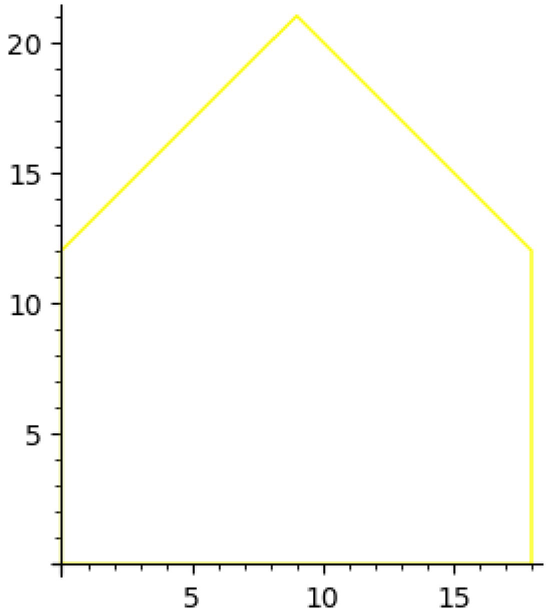
In [36]:
```
show(A,B,C,D)
```

$$\begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

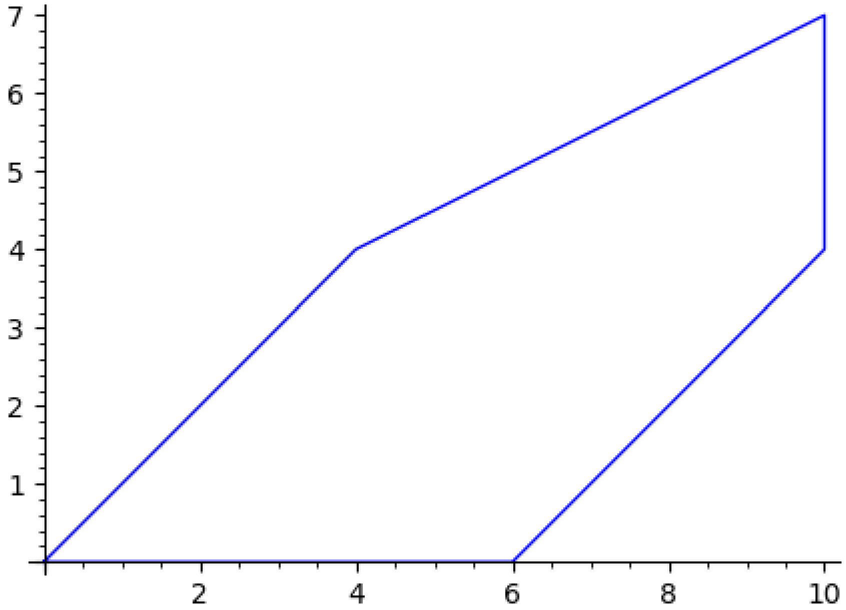
In [37]:
```
L1=list( [ [0,0], [6,0], [6,4], [3,7], [0,4],[0,0] ])
SL1=line(L1, color="red")
SL1.show(aspect_ratio=1, figsize=5)
```



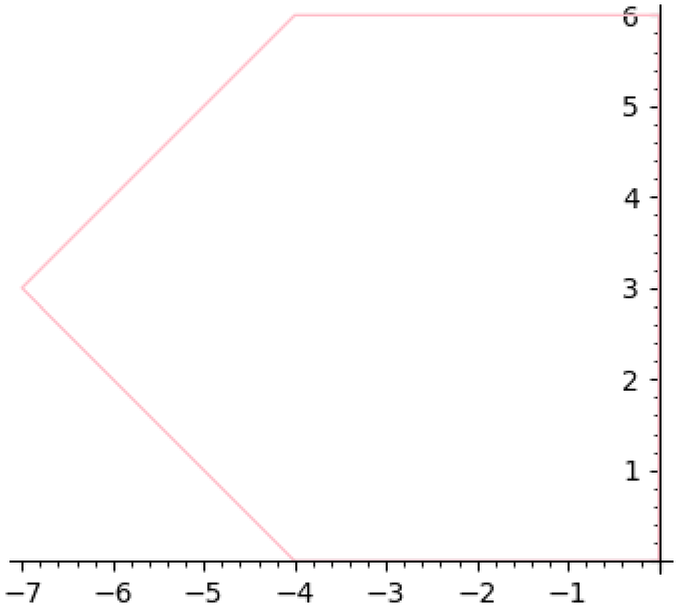
In [38]:
```
L2=matrix_transformation(A,L1)
SL2=line(L2,color="yellow")
SL2.show(aspect_ratio=1,figsize=5)
```



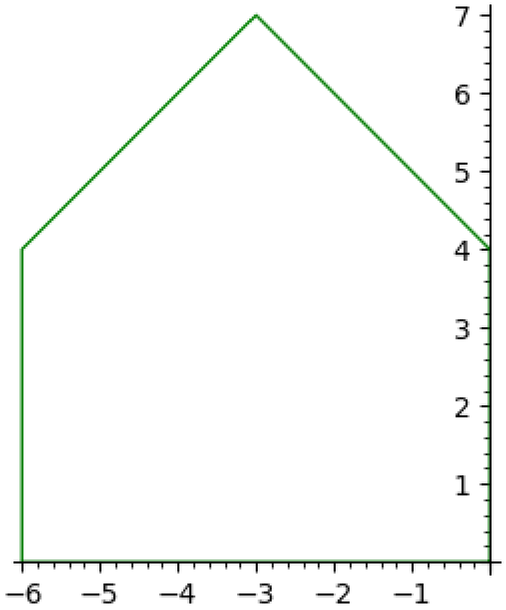
In [39]:
```
L3=matrix_transformation(B,L1)
SL3=line(L3,color="blue")
SL3.show(aspect_ratio=1,figsize=5)
```



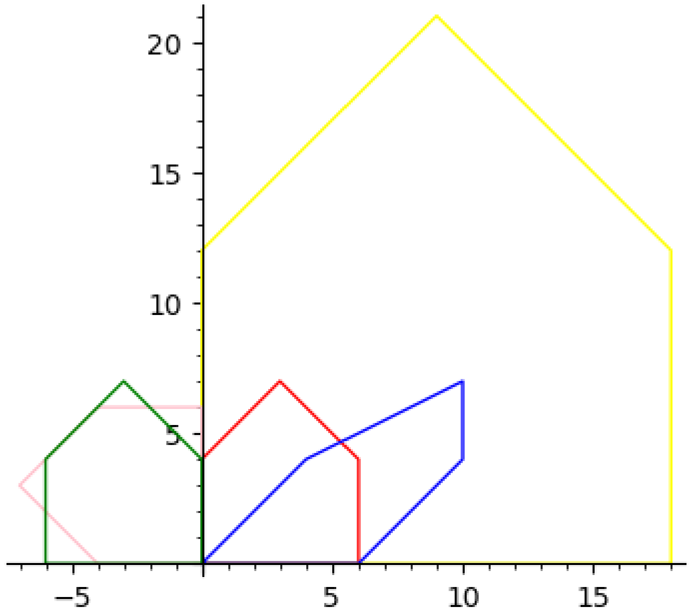
In [40]:
```
L4=matrix_transformation(C,L1)
SL4=line(L4,color="pink")
SL4.show(aspect_ratio=1,figsize=5)
```



In [41]:
```
L5=matrix_transformation(D,L1)
SL5=line(L5,color="green")
SL5.show(aspect_ratio=1,figsize=5)
```



In [42]:
```
(SL1+SL2+SL3+SL4+SL5).show(aspect_ratio=1,figsize=5)
```



2. Construct a shape by joining the points

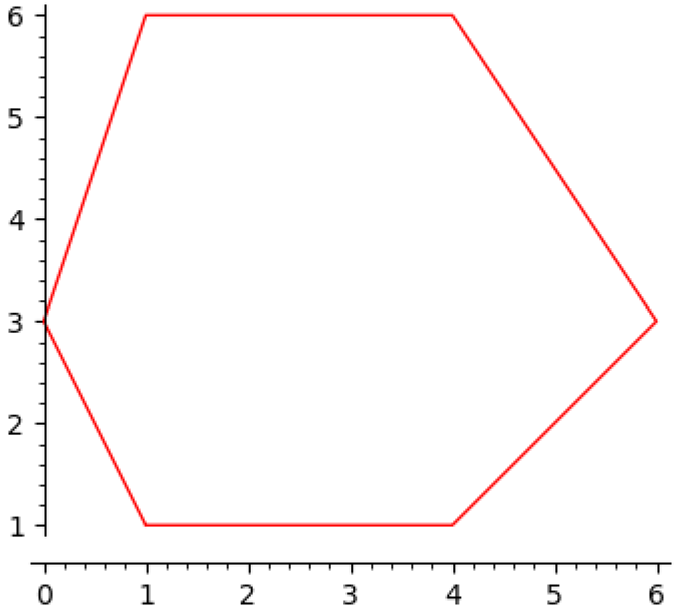
(1,1),(4,1),(6,3),(4,6),(1,6),(0,3),(1,1). Apply proper matrix transformations for: Expanding to thrice the given size. Shearing along the X-axis with a scale of 2. Rotating counter-clockwise by an angle of π/4.

In [43]:
```
A = matrix([[3,0],[0,3]])
B = matrix([[1,2],[0,1]])
C = matrix([[cos(pi/4), -sin(pi/4)],[sin(pi/4),  cos(pi/4)]])
```

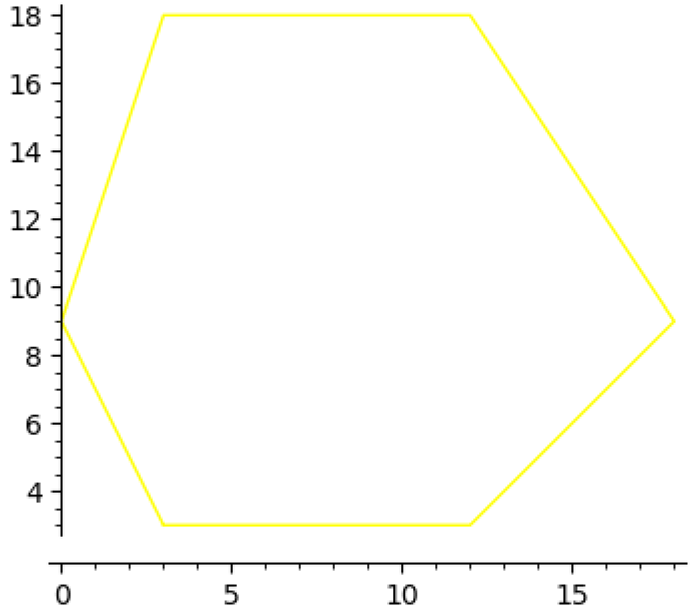
In [44]:
```
show(A,B,C)
```

$$\begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{2}\sqrt{2} & -\frac{1}{2}\sqrt{2} \\ \frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} \end{pmatrix}$$

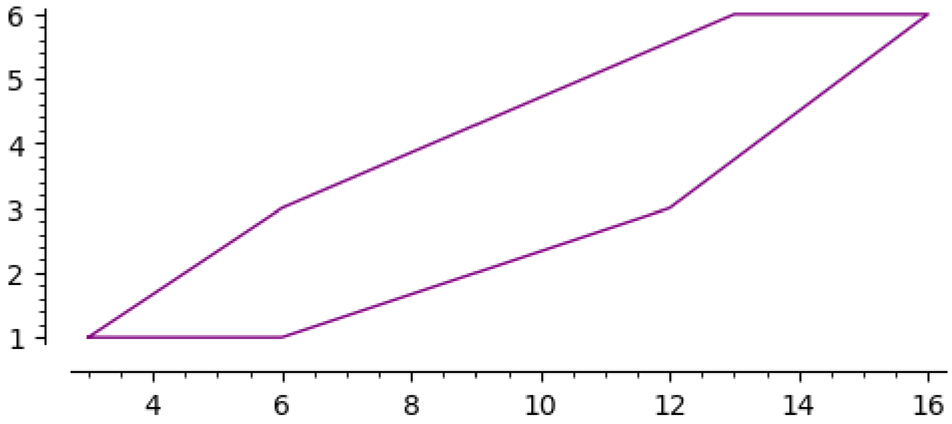
In [45]:
```
L1=list( [ [1,1], [4,1], [6,3], [4,6], [1,6],[0,3], [1,1] ])
SL1=line(L1, color="red")
SL1.show(aspect_ratio=1, figsize=5)
```



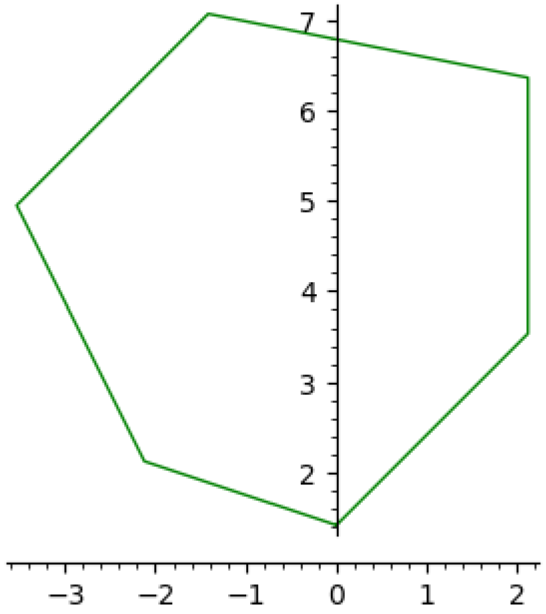
In [46]:
```
L2=matrix_transformation(A,L1)
SL2=line(L2,color="yellow")
SL2.show(aspect_ratio=1,figsize=5)
```



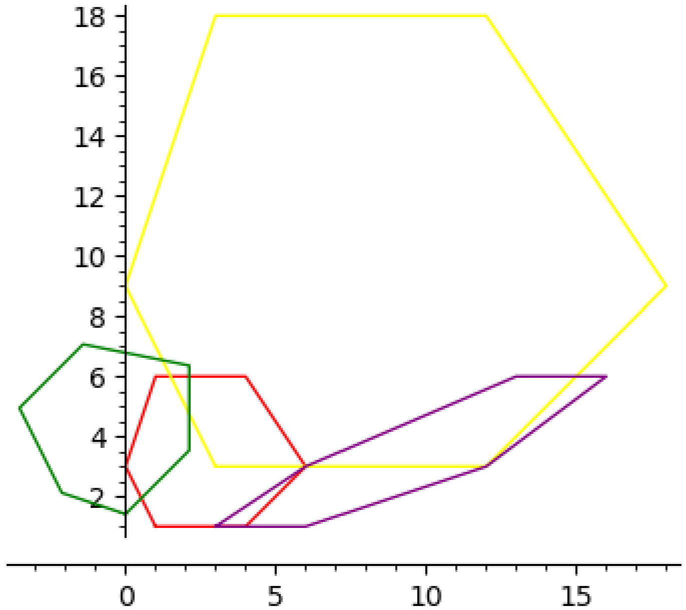
In [47]:
```
L3=matrix_transformation(B,L1)
SL3=line(L3,color="purple")
SL3.show(aspect_ratio=1,figsize=5)
```



In [48]:
```
L4=matrix_transformation(C,L1)
SL4=line(L4,color="green")
SL4.show(aspect_ratio=1,figsize=5)
```



In [49]:
```
(SL1+SL2+SL3+SL4).show(aspect_ratio=1,figsize=5)
```



1. Write the complete Singular value decomposition of the following matrices

(a)$\begin{pmatrix} 1 & 3 & 4 \end{pmatrix}$

$\begin{pmatrix} 2 & 1 & 9 \end{pmatrix}$

(b)$\begin{pmatrix} 1 \end{pmatrix}\begin{pmatrix} 2 \end{pmatrix}\begin{pmatrix} 4 \end{pmatrix}$

In [50]:
```
A = matrix(RDF, [[1,3,4], [2,1,9]])
show(A)
```

$$\begin{pmatrix} 1.0 & 3.0 & 4.0 \\ 2.0 & 1.0 & 9.0 \end{pmatrix}$$

In [51]:
```
U, S, V = A.SVD()
```

In [52]:
```
show(U)
show(S)
show(V)
```

$$\begin{pmatrix} -0.45248757199274725 & -0.8917707088664153 \\ -0.8917707088664153 & 0.4524875719927469 \end{pmatrix}$$

$$\begin{pmatrix} 10.33457997131582 & 0.0 & 0.0 \\ 0.0 & 2.2795738234323206 & 0.0 \end{pmatrix}$$

$$\begin{pmatrix} -0.2163637995866103 & 0.005792501643660683 & -0.9762956279494203 \\ -0.21764149400241908 & -0.9751053165102694 & 0.0424476359978014 \\ -0.951745179297925 & 0.22166657086289546 & 0.21223817998900435 \end{pmatrix}$$

In [53]:
```
U*S*(V.transpose())
```

Out[53]:
```
[0.9999999999999998 2.9999999999999987  4.000000000000001]
[ 1.999999999999996 0.9999999999999976  8.999999999999998]
```

In [54]:
```
B=matrix(CDF,3,1,[1,2,4])
```

In [55]:
```
show(B)
```

$$\begin{pmatrix} 1.0 \\ 2.0 \\ 4.0 \end{pmatrix}$$

In [56]:
```
U, S, V = A.SVD()
```

In [57]:
```
show(U)
show(S)
show(V)
```

$$\begin{pmatrix} -0.45248757199274725 & -0.8917707088664153 \\ -0.8917707088664153 & 0.4524875719927469 \end{pmatrix}$$

$$\begin{pmatrix} 10.33457997131582 & 0.0 & 0.0 \\ 0.0 & 2.2795738234323206 & 0.0 \end{pmatrix}$$

$$\begin{pmatrix} -0.2163637995866103 & 0.005792501643660683 & -0.9762956279494203 \\ -0.21764149400241908 & -0.9751053165102694 & 0.0424476359978014 \\ -0.951745179297925 & 0.22166657086289546 & 0.21223817998900435 \end{pmatrix}$$

In [58]:
```
U*S*(V.transpose())
```

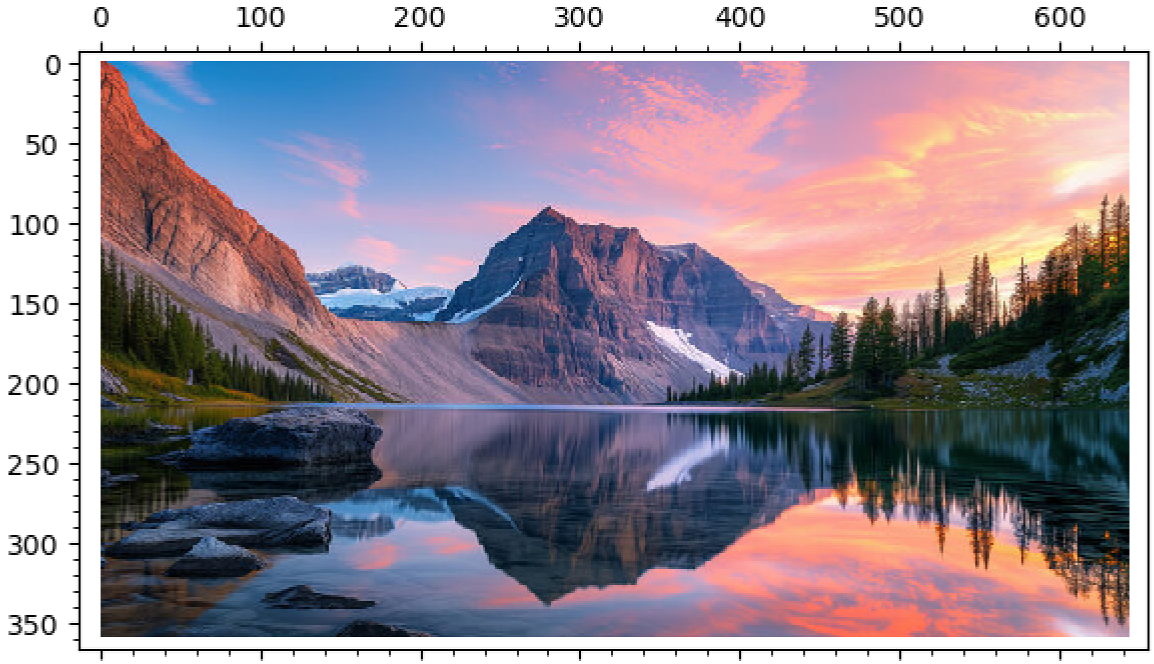
Out[58]:
```
[0.9999999999999998 2.9999999999999987  4.000000000000001]
[ 1.999999999999996 0.9999999999999976  8.999999999999998]
```

1. Take any two images of your choice and reduce the dimension by taking 50 singular values and 100 singular values. (Image should be strictly in png format)

In [59]:
```
from matplotlib.pyplot import imread
import pylab
import numpy as np
img=pylab.imread('img2.png')
```

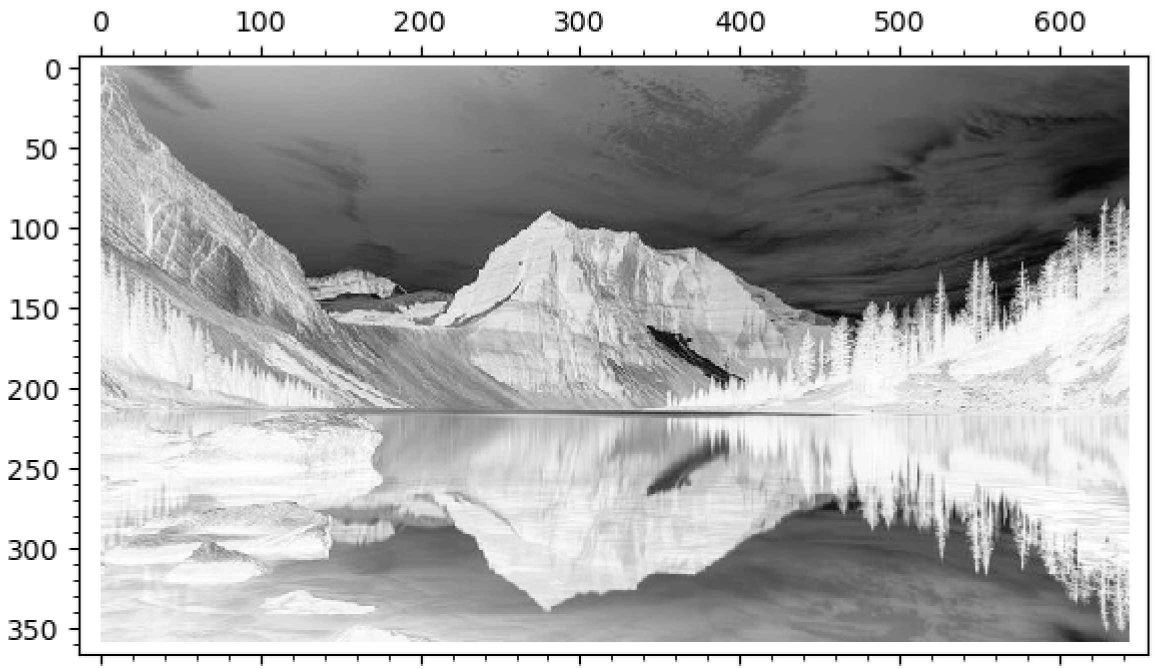
In [60]:
```
matrix_plot(img)
```

Out[60]:



In [61]:
```
gray=lambda rgb: np.dot(rgb[...,:3],[0.3,0.6,0.1])
G=gray(img)
matrix_plot(G)
```

Out[61]:



In [62]:
```
U,S,V=matrix(G).SVD()
```

In [63]:

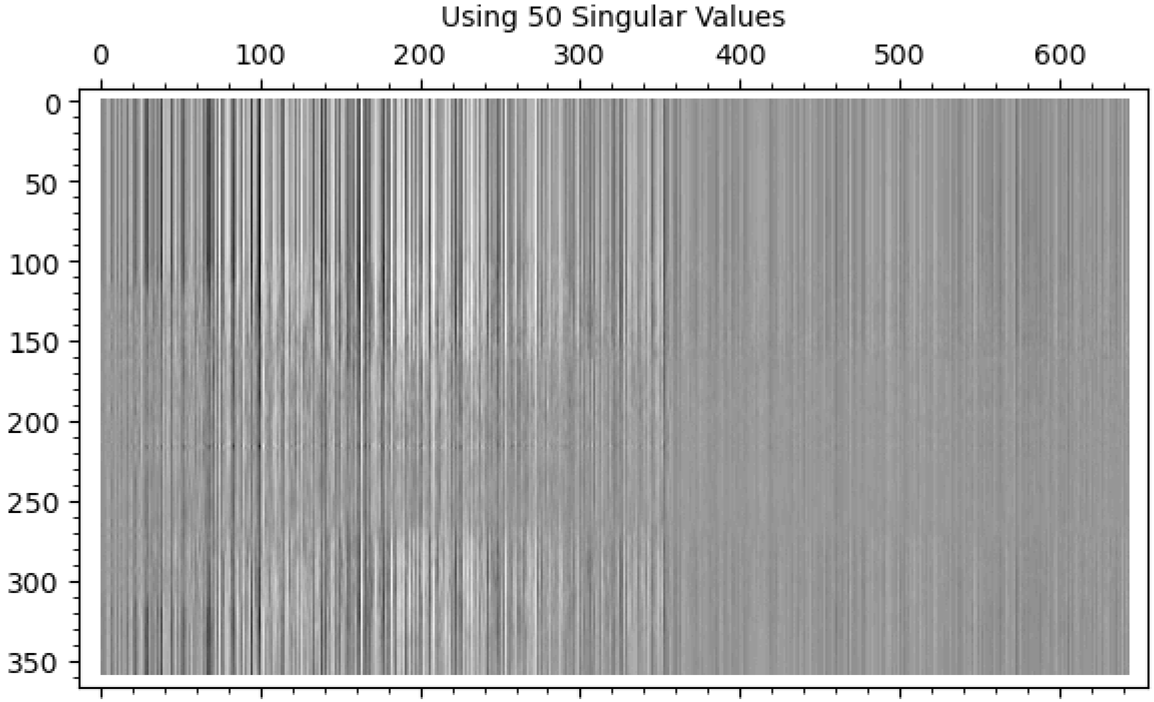
```
show(img)
```

```
[[[0.58431375 0.20784314 0.14509805]
  [0.64705884 0.23137255 0.14509805]
  [0.58431375 0.25882354 0.1882353 ]
  ...
  [0.8666667  0.7058824  0.7764706 ]
  [0.8666667  0.7058824  0.7764706 ]
  [0.8666667  0.7058824  0.7764706 ]]

 [[0.6156863  0.24705882 0.20784314]
  [0.6862745  0.2509804  0.2784314 ]
  [0.5921569  0.23137255 0.23529412]
  ...
  [0.87058824 0.7019608  0.76862746]
  [0.87058824 0.7019608  0.76862746]
  [0.87058824 0.7019608  0.76862746]]

 [[0.6313726  0.24705882 0.10196079]
  [0.5921569  0.30588236 0.23137255]
  [0.58431375 0.23529412 0.25882354]
  ...
  [0.8784314  0.7058824  0.75686276]
  [0.8784314  0.7058824  0.75686276]
  [0.8784314  0.7058824  0.75686276]]

 ...

 [[0.1254902  0.10980392 0.09803922]
  [0.13333334 0.11764706 0.10588235]
  [0.14509805 0.1254902  0.10980392]
  ...
  [0.69803923 0.5803922  0.68235296]
```

```
 [0.69411767 0.5764706  0.6784314 ]
 [0.7019608  0.57254905 0.6784314 ]]

[[0.15294118 0.13725491 0.1254902 ]
 [0.16078432 0.14901961 0.12941177]
 [0.17254902 0.16078432 0.14117648]
 ...
 [0.7176471  0.5882353  0.69411767]
 [0.7137255  0.58431375 0.6901961 ]
 [0.7137255  0.58431375 0.6901961 ]]

[[0.1764706  0.15686275 0.14117648]
 [0.18431373 0.17254902 0.15294118]
 [0.2        0.18431373 0.17254902]
 ...
 [0.73333335 0.6        0.7058824 ]
 [0.73333335 0.6        0.7058824 ]
 [0.7294118  0.59607846 0.7019608 ]]]
```

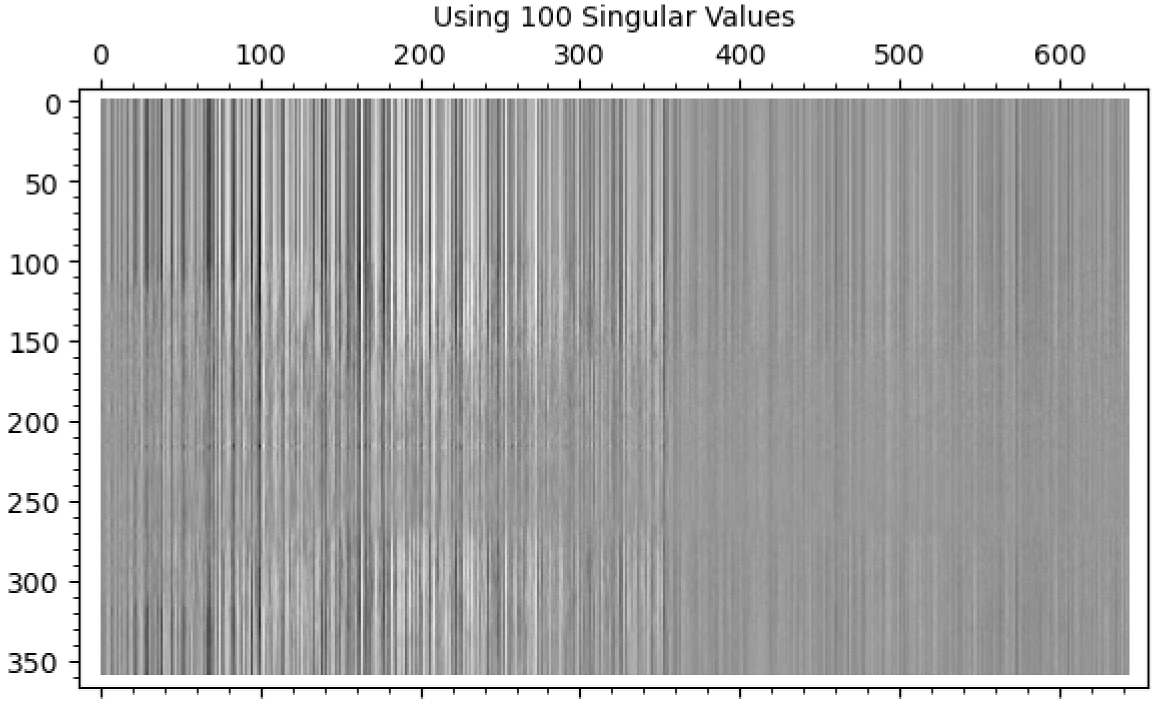
In [64]:
```
n = 50
A_50 = U[:,:n] * S[:n,:n] * V[:n,:]

matrix_plot(A_50, figsize=6, title="Using 50 Singular Values")
```

Out[64]:



In [65]:
```
n = 100
A_100 = U[:,:n] * S[:n,:n] * V[:n,:]

matrix_plot(A_100, figsize=6, title="Using 100 Singular Values")
```

Out[65]:



In [66]:
```
from matplotlib.pyplot import imread
import pylab
import numpy as np
img=pylab.imread('img3.png')
```

In [67]:
```
matrix_plot(img)
```

Out[67]:

In [68]:
```
U,S,V=matrix(G).SVD()
```

In [69]:

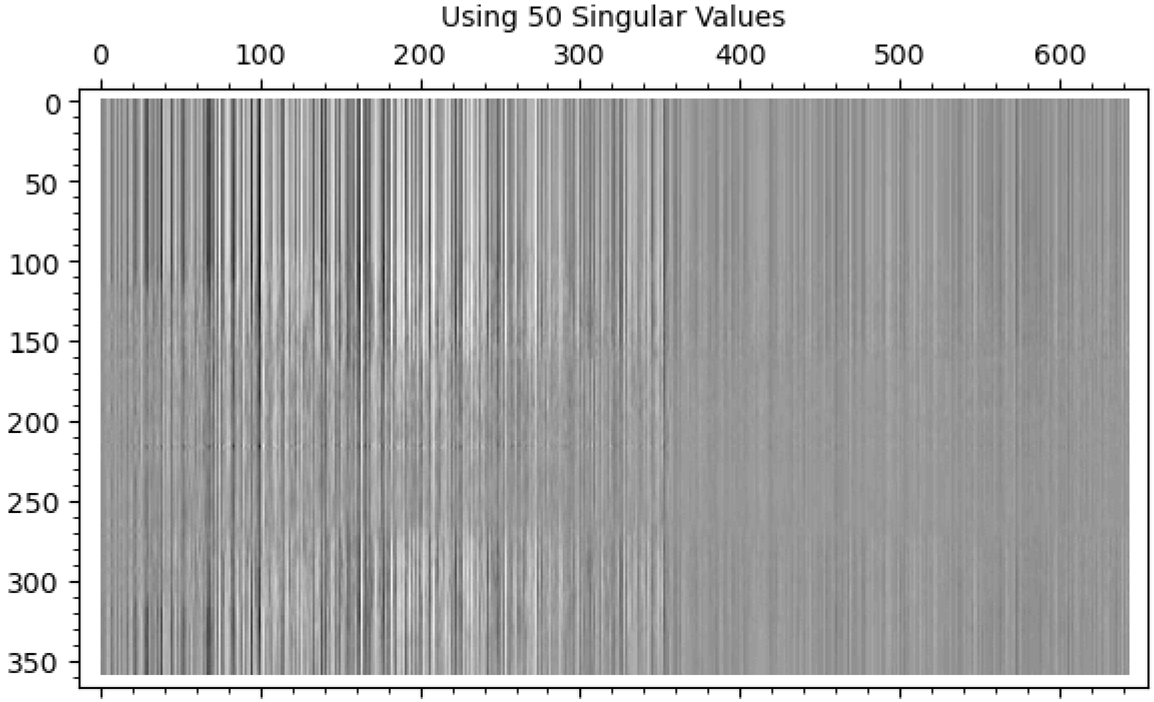
```
show(img)
```

```
[[[0.7490196  0.5019608  0.60784316]
  [0.7529412  0.5058824  0.6117647 ]
  [0.7607843  0.5137255  0.61960787]
  ...
  [0.93333334 0.7647059  0.7019608 ]
  [0.93333334 0.7647059  0.7019608 ]
  [0.92941177 0.7607843  0.69803923]]

 [[0.74509805 0.49803922 0.6039216 ]
  [0.7490196  0.5019608  0.60784316]
  [0.7607843  0.5137255  0.61960787]
  ...
  [0.93333334 0.7647059  0.7019608 ]
  [0.93333334 0.7647059  0.7019608 ]
  [0.93333334 0.7647059  0.7019608 ]]

 [[0.74509805 0.49803922 0.6039216 ]
  [0.7490196  0.5019608  0.60784316]
  [0.75686276 0.50980395 0.6156863 ]
  ...
  [0.9372549  0.76862746 0.7058824 ]
  [0.9372549  0.76862746 0.7058824 ]
  [0.9372549  0.76862746 0.7058824 ]]

 ...

 [[0.11372549 0.3529412  0.3254902 ]
  [0.07843138 0.31764707 0.2901961 ]
  [0.36078432 0.5647059  0.3764706 ]
  ...
  [0.9098039  0.8117647  0.58431375]
```

```
 [0.96862745 0.8627451  0.58431375]
 [0.9843137  0.8784314  0.6       ]]

[[0.13725491 0.3764706  0.34901962]
 [0.07058824 0.30980393 0.28235295]
 [0.2627451  0.46666667 0.2784314 ]
 ...
 [0.81960785 0.72156864 0.49411765]
 [0.92156863 0.8156863  0.5372549 ]
 [0.9647059  0.85882354 0.5803922 ]]

[[0.16078432 0.38039216 0.32941177]
 [0.12941177 0.34901962 0.29803923]
 [0.09411765 0.3372549  0.19215687]
 ...
 [0.7176471  0.65882355 0.49803922]
 [0.8745098  0.7764706  0.5176471 ]
 [0.9372549  0.8392157  0.5803922 ]]]
```

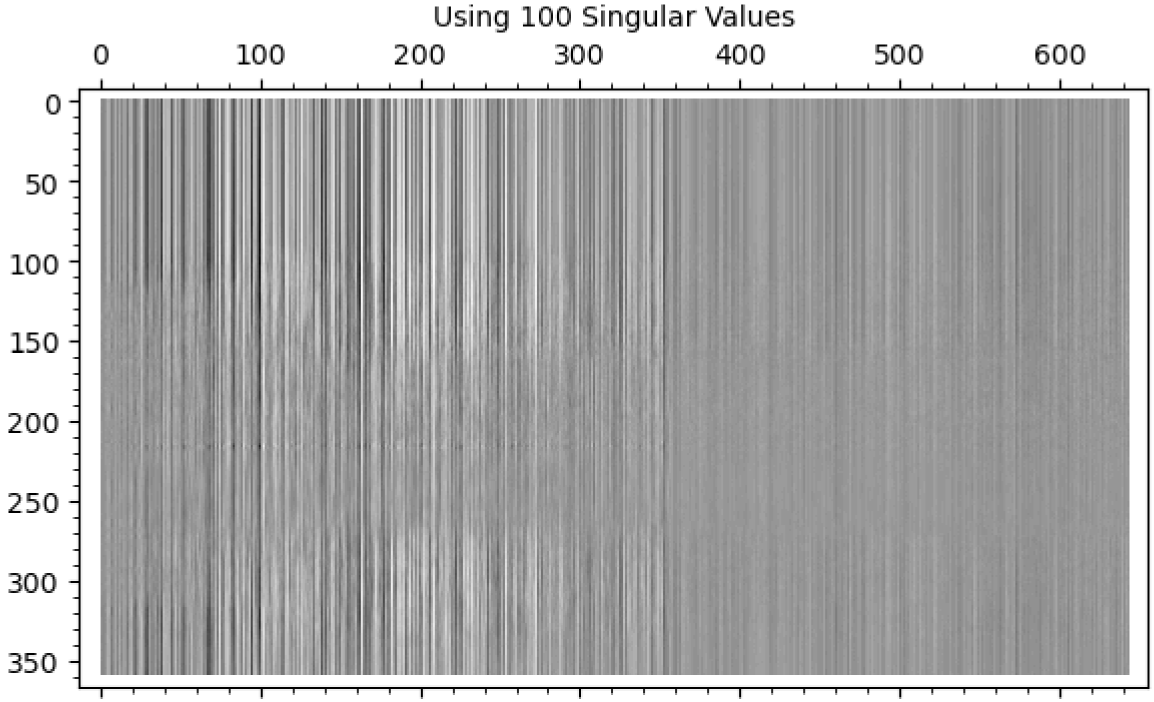
In [77]:
```
n = 50
A_50 = U[:,:n] * S[:n,:n] * V[:n,:]

matrix_plot(A_50, figsize=6, title="Using 50 Singular Values")
```

Out[77]:



In [71]:
```
n = 100
A_100 = U[:,:n] * S[:n,:n] * V[:n,:]

matrix_plot(A_100, figsize=6, title="Using 100 Singular Values")
```

Out[71]:



What have you Learned?

In this experiment, I learned how matrices are used as transformations in computer graphics. I understood how to construct shapes using coordinate points and apply different linear transformations such as scaling (expansion), shearing, rotation, and reflection using transformation matrices. I also learned the mathematical concept of Singular Value Decomposition (SVD) and how a matrix can be decomposed into three matrices U U, Σ Σ, and V T V T . I understood how SVD helps in dimensionality reduction and image compression by reconstructing images using a limited number of singular values. Through this experiment, I gained practical knowledge of implementing matrix operations and SVD using SageMath/Python, and I understood the connection between linear algebra concepts and real-world applications like computer graphics and image processing.

In [ ]:
```

```