Group 2:
— Armin Ettenhofer
— Atul Agarwal
— Johannes Spies

2.1 $A=\begin{bmatrix}0 & 3/5 & 4/5\\ -3/5 & 0 & 0\\ -4/5 & 0 & 0\end{bmatrix}$

1a)

- $\chi_A(\lambda)=\det(\lambda I-A)=\det\left(\begin{bmatrix}\lambda & -3/5 & -4/5\\ 3/5 & \lambda & 0\\ 4/5 & 0 & \lambda\end{bmatrix}\right)$

$=\lambda^3+0+0-\frac{4}{5}\cdot\lambda\cdot\left(-\frac{4}{5}\right)-0-\lambda\cdot\left(\frac{3}{5}\right)\left(-\frac{3}{5}\right)$

$=\lambda^3+\frac{16}{25}\lambda+\frac{9}{25}\lambda=\lambda^3+\lambda=\lambda(1+\lambda^2)=\lambda(\lambda-i)(\lambda+i)$

- eigenvalues of A are $\lambda=0$, $\lambda=\pm i$ (roots of $\chi_A(\lambda)$)
- proof for spectral decomp. $\Rightarrow$ $\exists U$ unitary: $A\cdot U=\operatorname{diag}(\lambda_1\cdots\lambda_n)A$

where $U=(u_1|\ldots|u_n)$ where u_j are eigenvectors of A

- Find $u_j\in\mathbb{C}^3$: $Au_j=\lambda_j u_j \quad \forall j\in\{1,2,3\}$
  - $\lambda_2=i$: $Au_2=iu_2 \Leftrightarrow Au_2-iIu_2=0 \Leftrightarrow (A-iI)u_2=0$

$\Leftrightarrow \left(\begin{bmatrix}0 & 3/5 & 4/5\\ -3/5 & 0 & 0\\ -4/5 & 0 & 0\end{bmatrix}-iI\right)u_2=0 \Leftrightarrow \begin{bmatrix}-i & 3/5 & 4/5\\ -3/5 & -i & 0\\ -4/5 & 0 & -i\end{bmatrix}u_2=0$

using Gauß-Jacobi: $\begin{bmatrix}-i & 3/5 & 4/5\\ -3/5 & -i & 0\\ -4/5 & 0 & -i\end{bmatrix} \begin{matrix} i\cdot\frac{3}{5}\\ i\cdot\frac{4}{5}\end{matrix} \sim \begin{bmatrix}-i & 3/5 & 4/5\\ 0 & -16/25\,i & 12/25\,i\\ 0 & 12/25\,i & -9/25\,i\end{bmatrix}\begin{matrix}\\ \cdot 25/4\\ \cdot 25/3\end{matrix}$

using: $-\frac{3}{5}+\underbrace{(-i)i}_{+1}\frac{3}{5}=0$, $-\frac{4}{5}+(-i)i\cdot\frac{4}{5}=0$,

$\sim\begin{bmatrix}-i & 3/5 & 4/5\\ 0 & -4i & 3i\\ 0 & 4i & -3i\end{bmatrix}\begin{matrix}\cdot 5\\ \\ \end{matrix} \underset{1}{\sim} \begin{bmatrix}-5i & 3 & 4\\ 0 & -4i & 3i\\ 0 & 0 & 0\end{bmatrix}\begin{matrix}\left(\frac{3}{4i}\right)\\ \cdot i\end{matrix} \sim \begin{bmatrix}-5i & 0 & 25/4\\ 0 & -4 & 3\\ 0 & 0 & 0\end{bmatrix}\cdot\frac{4}{5} \sim \begin{bmatrix}-4i & 0 & 5\\ 0 & -4 & 3\\ 0 & 0 & 0\end{bmatrix}$

$\Leftrightarrow -4ix_1+5x_3=0 \wedge -4x_2+3x_3=0 \Leftrightarrow 4ix_1=5x_3 \wedge 4x_2=3x_3$

$\Leftrightarrow x_1=-\frac{5}{4}i x_3 \wedge x_2=\frac{3}{4}x_3$

$\Leftrightarrow \mathbb{L}=\left\{\left(-\frac{5}{4}ix_3 \;\; \frac{3}{4}x_3 \;\; x_3\right)^T \mid x_3\in\mathbb{C}\right\}$

$\Rightarrow$ choose $v_2=\left[-\frac{5}{4}i \;\; \frac{3}{4} \;\; i\right]^T$

  - $\lambda_3=-i$: $Au_3=\lambda_3u_3 \Leftrightarrow Av_3=-iv_3 \Leftrightarrow Av_3+iv_3=0 \Leftrightarrow (A+iI)v_3=0$

let $A+iI=\begin{bmatrix}i & 3/5 & 4/5\\ -3/5 & i & 0\\ -4/5 & 0 & i\end{bmatrix}$ ← solve LGS:

$$\begin{bmatrix} i & 3/5 & 4/5 \\ -3/5 & i & 0 \\ -4/5 & 0 & i \end{bmatrix} \begin{matrix} \\ (-\frac{3}{5}i) \\ (-\frac{4}{5}i) \end{matrix} \sim \begin{bmatrix} i & 3/5 & 4/5 \\ 0 & 16/25\, i & -12/25\, i \\ 0 & -12/25\, i & 9/25\, i \end{bmatrix} \begin{matrix} \\ \cdot 25/4 \\ \cdot 25/3 \end{matrix} \sim \begin{bmatrix} 5i & 3 & 4 \\ 0 & 4i & -3i \\ 0 & -4i & 3i \end{bmatrix} \begin{matrix} (-\frac{3i}{4i}) \\ \\ \cdot 1 \end{matrix} \begin{bmatrix} 5i & 0 & 25/4 \\ 0 & 4i & -3i \\ 0 & 0 & 0 \end{bmatrix} \cdot \frac{4}{5}$$

$$\sim \begin{bmatrix} 4i & 0 & 5 \\ 0 & 4i & -3i \\ 0 & 0 & 0 \end{bmatrix} \Leftrightarrow \begin{matrix} 4ix_1 + 5x_3 = 0 \\ \wedge\, 4ix_2 - 3ix_3 = 0 \end{matrix} \Leftrightarrow \begin{matrix} 4ix_1 = -5x_3 \\ \wedge\, 4ix_2 = 3ix_3 \end{matrix} \Leftrightarrow \begin{matrix} x_1 = -\frac{5}{4i}x_3 \\ \wedge\, x_2 = 3/4\, x_3 \end{matrix}$$

$$\Leftrightarrow x_1 = \frac{5}{4}ix_3 \wedge x_2 = \frac{3}{4}x_3 \Leftrightarrow \mathbb{L} = \{(\tfrac{5}{4}ix_3 \;\; \tfrac{3}{4}x_3 \;\; x_3) \mid x_3 \in \mathbb{C}\}$$

$\Rightarrow$ choose $u_3 = (5/4\, i \;\; 3/4 \;\; 1)^T$

- $\lambda_1 = 0$: $Au_1 = \lambda_1 u_1 \Leftrightarrow Au_1 = 0$

$$\begin{bmatrix} 0 & 3/5 & 4/5 \\ -3/5 & 0 & 0 \\ -4/5 & 0 & 0 \end{bmatrix} \begin{matrix} \\ \\ (\frac{5}{3} \cdot \frac{4}{5}) \end{matrix} \sim \begin{bmatrix} 0 & 3/5 & 4/5 \\ -3/5 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{matrix} \cdot 5 \\ \cdot (-\frac{5}{3}) \\ \\ \end{matrix} \sim \begin{bmatrix} 0 & 3 & 4 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\Leftrightarrow 3x_2 + 4x_3 = 0 \wedge x_1 = 0 \Leftrightarrow x_2 = -\frac{4}{3}x_3$$

$$\Leftrightarrow \mathbb{L} = \{(0 \;\; -\tfrac{4}{3}x_3 \;\; x_3)^T \mid x_3 \in \mathbb{C}\}$$

- choose unitary matrix $U = (u_1 \mid u_2 \mid u_3)^T$ for Spectral Decomp.:

$$\begin{array}{cc} & \\ U = & \begin{pmatrix} 0 & -5/4\, i & 5/4\, i \\ -4/3 & 3/4 & 3/4 \\ 1 & 1 & 1 \end{pmatrix} \\ & \begin{matrix} \uparrow & \quad \uparrow & \quad \uparrow \\ \lambda = 0 & \lambda = i & \lambda = -i \end{matrix} \end{array}$$

(b) • A normal $\Rightarrow$ A is unitarily diagonalizable (Spectral decomp. Thm.):

• $\Rightarrow \exists U \in \mathbb{C}^{n\times n}$: $A \cdot U = U \cdot \mathrm{diag}(\lambda_1 \cdots \lambda_n)$ | $\cdot U^\dagger$

$\Leftrightarrow A \cdot UU^\dagger = U \mathrm{diag}(\lambda_1 \cdots \lambda_n) U^\dagger$ | • Def. unitary: $UU^\dagger = I$, • $\forall A \in \mathbb{C}^{n\times n}$: $A \cdot I = A$ w/o Pf.

$\Leftrightarrow A = U \mathrm{diag}(\lambda_1 \cdots \lambda_n) U^\dagger$

• $\mathrm{tr}[A] = \mathrm{tr}[U \mathrm{diag}(\lambda_1 \cdots \lambda_n) U^\dagger]$ | Hint

$= \mathrm{tr}[\mathrm{diag}(\lambda_1 \cdots \lambda_n) U^\dagger U]$ | Def. unitary $UU^\dagger = I \overset{U \text{ normal}}{=} U^\dagger U$

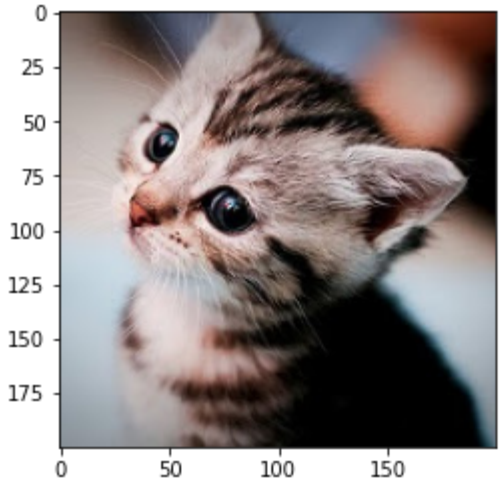
$= \mathrm{tr}[\mathrm{diag}(\lambda_1 \cdots \lambda_n) \cdot I]$ | $\forall A \in \mathbb{C}^{n\times n}$: $IA = A$ w/o Pf.

$= \mathrm{tr}[\mathrm{diag}(\lambda_1 \cdots \lambda_n)]$ | By def. matrix trace

$= \sum_{j=1}^{n} \lambda_j$ $\square$

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
# read image from disk
imgc = plt.imread("kitten.jpg")
plt.imshow(imgc);
```



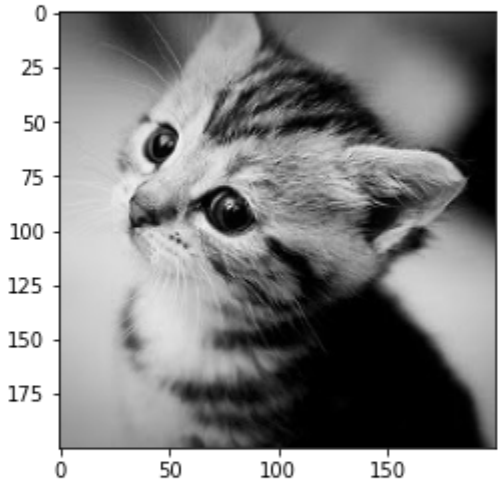
In [4]:
```
imgc.shape
```

Out[4]:
```
(200, 200, 3)
```

In [5]:
```
# convert to grayscale and [0, 1] value range
img = np.mean(imgc, axis=2) / 255.0
img.shape
```

Out[5]:
```
(200, 200)
```

In [6]:
```
plt.imshow(img, cmap="gray");
```

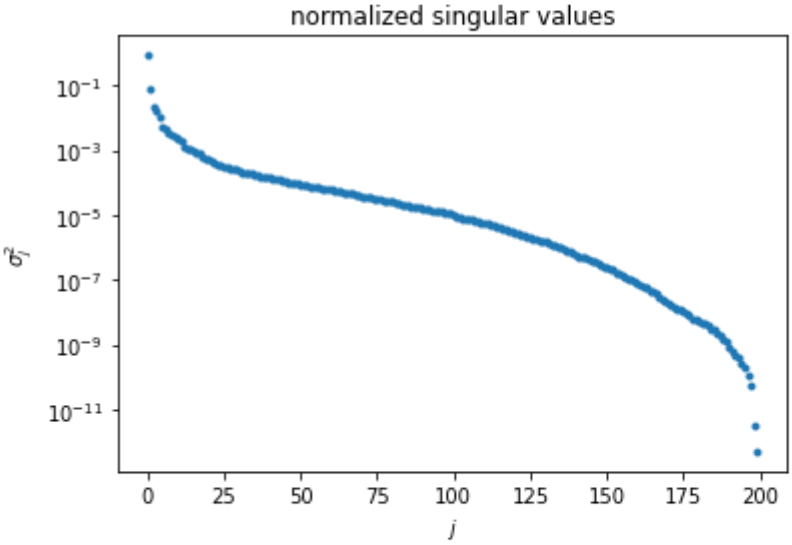


In [7]:
```
# singular value decomposition
u, s, vh = np.linalg.svd(img)
```

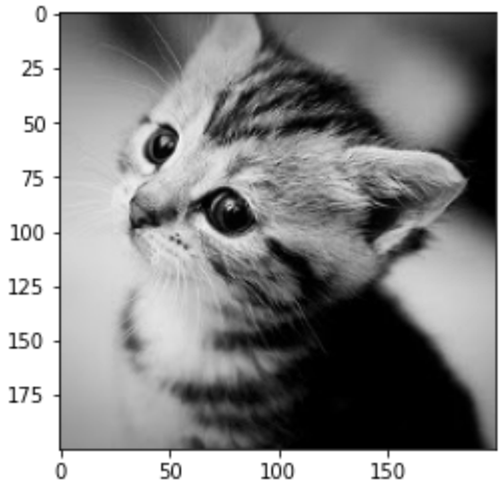
In [8]:
```
# check
np.allclose(img, (u * s) @ vh)
```

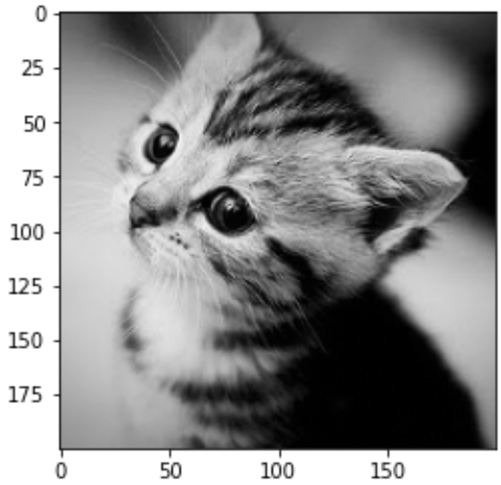
Out[8]:
```
True
```

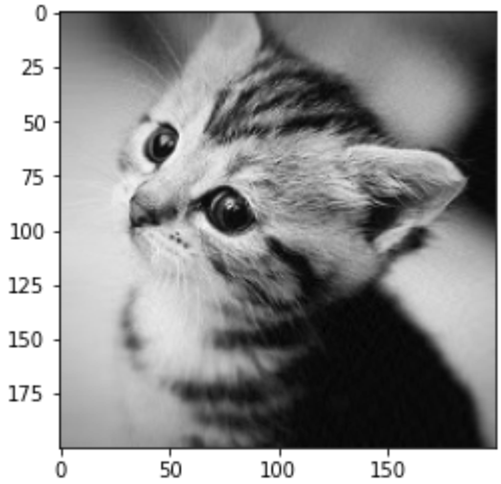
In [9]:
```
plt.semilogy(s**2 / np.sum(s**2), '.')
plt.ylabel("$\\sigma_j^2$")
plt.xlabel("$j$")
plt.title("normalized singular values");
```

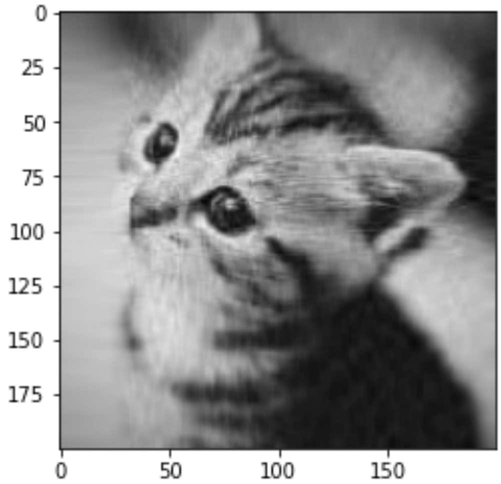


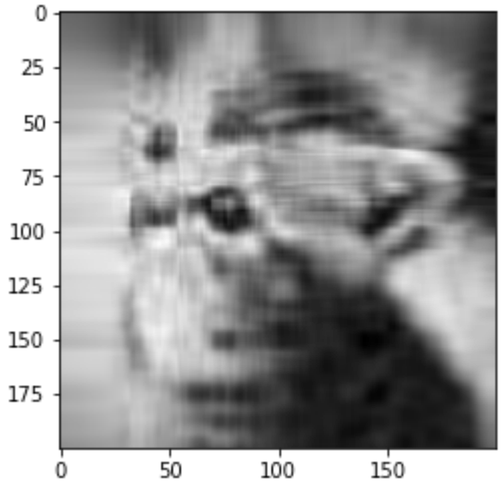
In [10]:
```
χlist = [len(s), 150, 75, 25, 10]

# TODO: Compute and plot the SVD-compressed approximation of `img` for all
# Hint: `u[:, :χ]` selects the leading `χ` columns of the matrix `u`.
for x in χlist:
    u_tilde = u[:, :x]
    s_tilde = s[:x]

    # manually create vh_tilde from v
    v = np.conj(vh.T)
    v_tilde = v[:, :x]
    vh_tilde = np.conj(v_tilde).T

    # reconstruct image
    img_tilde = u_tilde * s_tilde @ vh_tilde

    # show image
    fig, ax = plt.subplots()
    ax.imshow(img_tilde, cmap="gray")
```




0
25
50
75
100
125
150
175
0
50
100
150

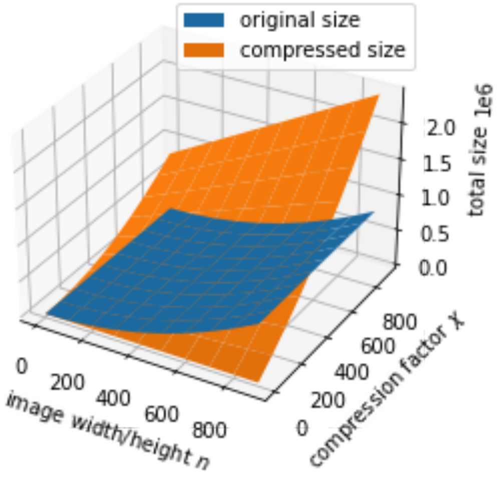

0
25
50
75
100
125
150
175
0
50
100
150


0
25
50
75
100
125
150
175
0
50
100
150


0
25
50
75
100
125
150
175
0
50
100
150

In [12]:
```
# bonus: show how compression level affects stored size
n = np.arange(0, 1000, 100)
chi = np.arange(0, 1000, 100)
n, chi = np.meshgrid(n, chi)
original = n*n
compressed = 2*n*chi + chi*chi
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
orig_surf = ax.plot_surface(n, chi, original, label="original size")
comp_surf = ax.plot_surface(n, chi, compressed, label="compressed size")

# create legend
orig_surf._edgecolors2d = orig_surf._edgecolor3d
orig_surf._facecolors2d = orig_surf._facecolor3d
comp_surf._edgecolors2d = comp_surf._edgecolor3d
comp_surf._facecolors2d = comp_surf._facecolor3d
ax.legend()

ax.set_xlabel("image width/height $n$")
ax.set_ylabel("compression factor $\\chi$")
ax.set_zlabel("total size")
```

Out[12]: Text(0.5, 0, 'total size')



In [14]:
```
# exercise 2.2a
def spectral_decomp(A):
    # eigenvalues w and eigenvectors v
    w, v = np.linalg.eig(A)
    U = np.array(v)

    # calculate spectral decomposition: A*U = U*diag(lambda1 ... lambda3)
    print("Spectral Decomposition:")
    lhs = A@U
    print(f"A*U={lhs}")
    rhs = U@np.diag(w)
    print(f"U*d={rhs}")
    assert np.allclose(lhs, rhs)

def singular_val_decomp(A):
    print("Singular Value Decomposition:")
    u, s, vh = np.linalg.svd(A)
    print(f"U={u}")
    print(f"S=diag({s})")
    print(f"V^t={vh}")
    A_prime = u*s@vh
    print(f"U*S*V^t={A_prime}")
    assert np.allclose(A, A_prime)
```

```
# matrices from T2 and H2.1
A1 = np.array([[1, -1j, 0], [1j, 1, 0], [0, 0, 1j]])
A2 = np.array([[0, 3/5, 4/5], [-3/5, 0, 0], [-4/5, 0, 0]])
print(f"A={A1}")
spectral_decomp(A1)
singular_val_decomp(A1)
print(f"\nA={A2}")
spectral_decomp(A2)
singular_val_decomp(A2)
```

```
A=[[ 1.+0.j -0.-1.j  0.+0.j]
 [ 0.+1.j  1.+0.j  0.+0.j]
 [ 0.+0.j  0.+0.j  0.+1.j]]
Spectral Decomposition:
A*U=[[0.00000000e+00-1.41421356e+00j 2.22044605e-16+0.00000000e+00j
  0.00000000e+00+0.00000000e+00j]
 [1.41421356e+00+0.00000000e+00j 0.00000000e+00+2.22044605e-16j
  0.00000000e+00+0.00000000e+00j]
 [0.00000000e+00+0.00000000e+00j 0.00000000e+00+0.00000000e+00j
  0.00000000e+00+1.00000000e+00j]]
U*d=[[0.        -1.41421356j 0.        +0.j         0.        +0.j        ]
 [1.41421356+0.j         0.        +0.j         0.        +0.j        ]
 [0.        +0.j         0.        +0.j         0.        +1.j        ]]
Singular Value Decomposition:
U=[[-0.70710678+0.j          0.        +0.j          0.        +0.70710678
j]
 [ 0.        -0.70710678j  0.        +0.j          0.70710678+0.j        ]
 [ 0.        +0.j          0.        -1.j          0.        +0.j        ]]
S=diag([2. 1. 0.])
V^t=[[-0.70710678+0.j          0.        +0.70710678j -0.        +0.j
]
 [-0.        +0.j          0.        +0.j          -1.        +0.j        ]
 [-0.70710678+0.j          0.        -0.70710678j  0.        +0.j        ]]
U*S*V^t=[[1.+0.j 0.-1.j 0.+0.j]
 [0.+1.j 1.+0.j 0.+0.j]
 [0.+0.j 0.+0.j 0.+1.j]]

A=[[ 0.   0.6  0.8]
 [-0.6  0.   0. ]
 [-0.8  0.   0. ]]
Spectral Decomposition:
A*U=[[ 0.00000000e+00+0.70710678j  0.00000000e+00-0.70710678j
   1.15463195e-16+0.j        ]
 [-4.24264069e-01+0.j         -4.24264069e-01+0.j
   0.00000000e+00+0.j        ]
 [-5.65685425e-01+0.j         -5.65685425e-01+0.j
   0.00000000e+00+0.j        ]]
U*d=[[ 0.        +0.70710678j  0.        -0.70710678j  0.        +0.j
]
 [-0.42426407+0.j         -0.42426407+0.j          0.        +0.j        ]
 [-0.56568542+0.j         -0.56568542+0.j          0.        +0.j        ]]
Singular Value Decomposition:
U=[[-1.00000000e+00  0.00000000e+00 -6.66133815e-18]
 [ 0.00000000e+00  6.00000000e-01 -8.00000000e-01]
 [ 1.11022302e-16  8.00000000e-01  6.00000000e-01]]
S=diag([1. 1. 0.])
V^t=[[-0.  -0.6 -0.8]
 [-1.  -0.  -0. ]
 [ 0.  -0.8  0.6]]
U*S*V^t=[[ 0.00000000e+00  6.00000000e-01  8.00000000e-01]
 [-6.00000000e-01  0.00000000e+00  0.00000000e+00]
 [-8.00000000e-01 -6.66133815e-17 -8.88178420e-17]]
```

In [ ]: