```
import matplotlib.pyplot as plt
import numpy as np
```

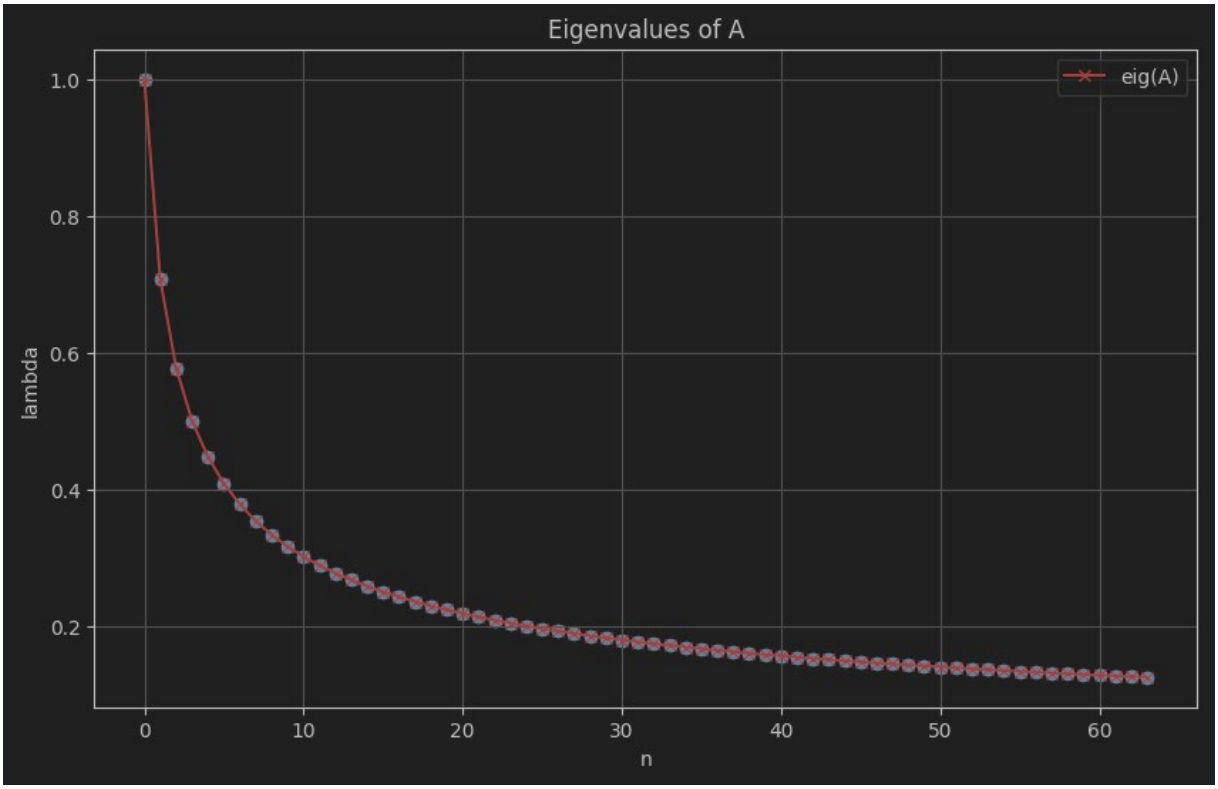
```
n = 64
A = np.zeros((n, n))

for j in range(1, n + 1):
    A[j - 1, j - 1] = 1 / np.sqrt(j)
    if j < n:
        A[j - 1, j] = 1 / np.sqrt(j)

A
```

```
array([[1.        , 1.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.70710678, 0.70710678, ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.57735027, ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 0.12700013, 0.12700013,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.12598816,
        0.12598816],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.125     ]])
```

```
lambda_A= np.linalg.eigvals(A)

lambda_A
```

```
array([1.        , 0.70710678, 0.57735027, 0.5       , 0.4472136 ,
       0.40824829, 0.37796447, 0.35355339, 0.33333333, 0.31622777,
       0.30151134, 0.28867513, 0.2773501 , 0.26726124, 0.25819889,
       0.25      , 0.24253563, 0.23570226, 0.22941573, 0.2236068 ,
       0.21821789, 0.21320072, 0.20851441, 0.20412415, 0.2       ,
       0.19611614, 0.19245009, 0.18898224, 0.18569534, 0.18257419,
       0.1796053 , 0.1767767 , 0.17407766, 0.17149859, 0.16903085,
       0.16666667, 0.16439899, 0.16222142, 0.16012815, 0.15811388,
       0.15617376, 0.15430335, 0.15249857, 0.15075567, 0.1490712 ,
       0.14744196, 0.14586499, 0.14433757, 0.14285714, 0.14142136,
       0.14002801, 0.13867505, 0.13736056, 0.13608276, 0.13483997,
       0.13363062, 0.13245324, 0.13130643, 0.13018891, 0.12909944,
       0.12803688, 0.12700013, 0.12598816, 0.125     ])
```

```
plt.figure(figsize=(10, 6))
plt.plot(range(0,n), lambda_A, 'x-', color='r', label='eig(A)')
plt.scatter(range(0,n), lambda_A)
plt.title('Eigenvalues of A')
plt.xlabel('n')
plt.ylabel('lambda')
plt.legend()
plt.grid(True)
plt.show()
```



The eigenvalues of A show a decreasing trend, aligning with the structure of AA, where diagonal elements constructed as 1/j decrease as j increases. Sub-diagonal elements further contribute to this trend.

```
def arnoldi_iteration(A, b, k):
    n = A.shape[0]
    H = np.zeros((k + 1, k))
    Q = np.zeros((n, k + 1))
    Q[:, 0] = b / np.linalg.norm(b)

    for j in range(k):
        v = A @ Q[:, j]
        for i in range(j + 1):
            H[i, j] = np.dot(Q[:, i].conj().T, v)
            v = v - H[i, j] * Q[:, i]
        H[j + 1, j] = np.linalg.norm(v)
        if H[j + 1, j] != 0 and j + 1 < k:
            Q[:, j + 1] = v / H[j + 1, j]

    return H, Q
```

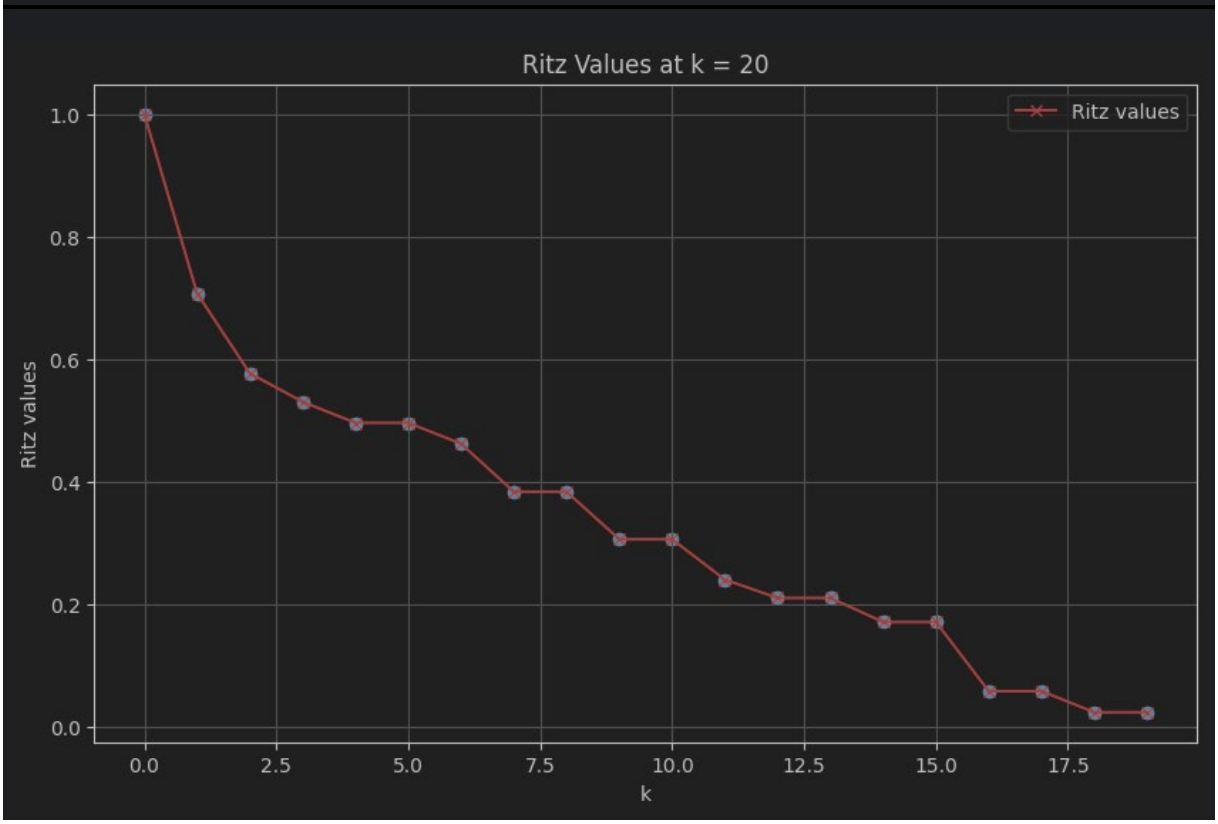
```
b = np.random.rand(n)
k = 20
```

```
H, Q = arnoldi_iteration(A, b, k)
```

```
ritz_values = np.linalg.eigvals(H[:k, :k])

ritz_values
```

```
array([1.         +0.j         , 0.70710524+0.j         ,
       0.57674109+0.j         , 0.530115  +0.j         ,
       0.49582015+0.07379503j, 0.49582015-0.07379503j,
       0.46175771+0.j         , 0.38342785+0.1169096j ,
       0.38342785-0.1169096j , 0.30572866+0.16873797j,
       0.30572866-0.16873797j, 0.23983276+0.j         ,
       0.2098191 +0.13960559j, 0.2098191 -0.13960559j,
       0.17056161+0.12563091j, 0.17056161-0.12563091j,
       0.05770428+0.10010078j, 0.05770428-0.10010078j,
       0.02325112+0.0507624j , 0.02325112-0.0507624j ])
```

```
plt.figure(figsize=(10, 6))
plt.plot(range(0,k), ritz_values, 'x-', color='r', label='Ritz values')
plt.scatter(range(0,k), ritz_values)
plt.title(f'Ritz Values at k = {k}')
plt.xlabel('k')
plt.ylabel('Ritz values')
plt.legend()
plt.grid(True)
plt.show()
```



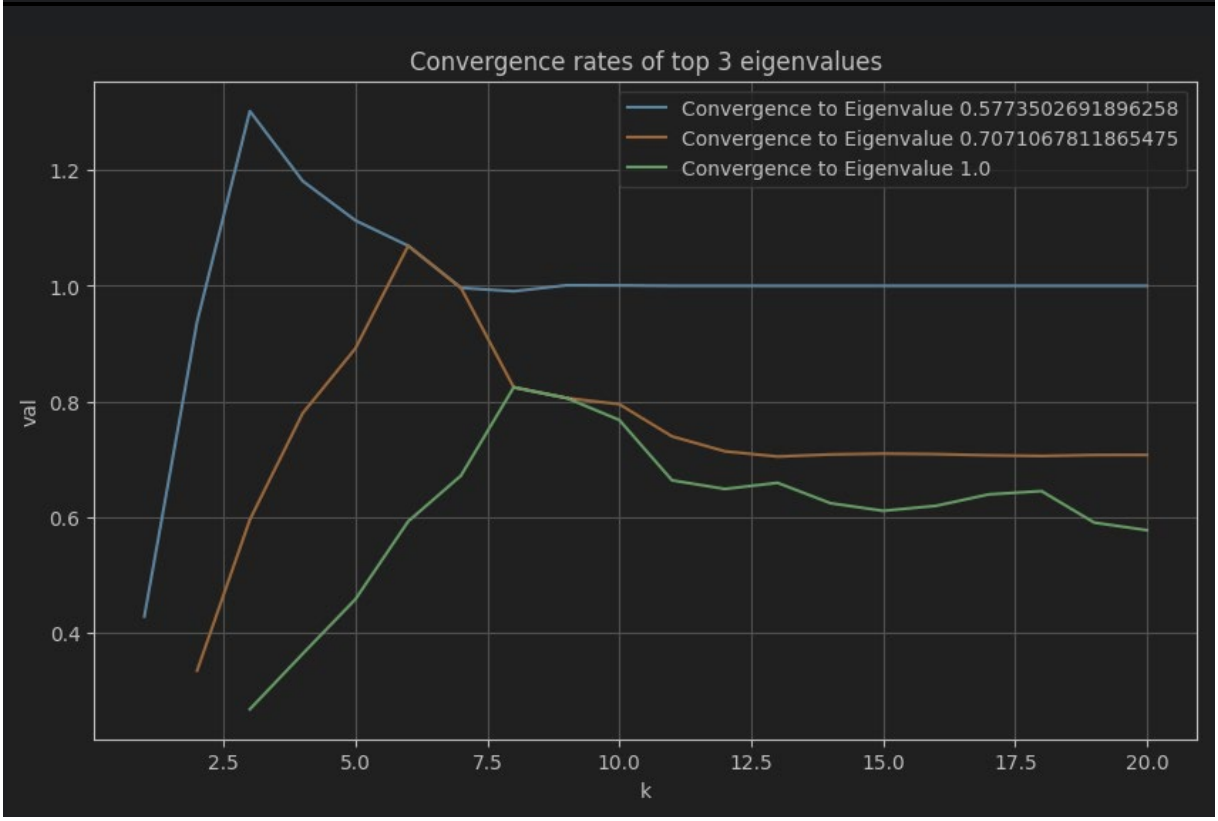
The Arnoldi iteration method approximates the eigenvalues, producing (complex) Ritz values that reflect the structure of A, involving complex eigenvalues due to the upper Hessenberg form during iteration.

```
largest_eigenvalues = np.sort(np.abs(lambda_A))[-3:]

largest_eigenvalues
```

```
array([0.57735027, 0.70710678, 1.        ])
```

```
convergence_rates = {i: [] for i in range(3)}
iteration_steps = list(range(1, 21))

for k in iteration_steps:
    H, Q = arnoldi_iteration(A, b, k)
    ritz_values = np.linalg.eigvals(H[:k, :k])
    sorted_ritz_values = np.sort(np.abs(ritz_values))
    for i in range(3):
        if i < len(sorted_ritz_values):
            convergence_rates[i].append(sorted_ritz_values[-(i+1)])
        else:
            convergence_rates[i].append(np.nan)
```

```
plt.figure(figsize=(10, 6))
for i in range(3):
    plt.plot(iteration_steps, convergence_rates[i], label=f'Convergence to
Eigenvalue {largest_eigenvalues[i]}')
plt.title('Convergence rates of top 3 eigenvalues')
plt.xlabel('k')
plt.ylabel('val')
plt.legend()
plt.grid(True)
plt.show()
```



The largest eigenvalue (1.0) converges quickly, stabilizing around k = 7.5, while the second largest (approx. 0.707) converges more slowly, stabilizing around k=12.5. The third one (approx. 0.577) is noisier and does not converge at all within the 20 epochs range, even though we see that the curve floats around 0.6.

Basically, these what these results confirm is that we observe rapid decay of the spectrum indicated by the largest eigenvalue of 1 and the non-rapid decay when approximating smaller eigenvalues with Arnoldi iteration. This is due to the spectral gap.