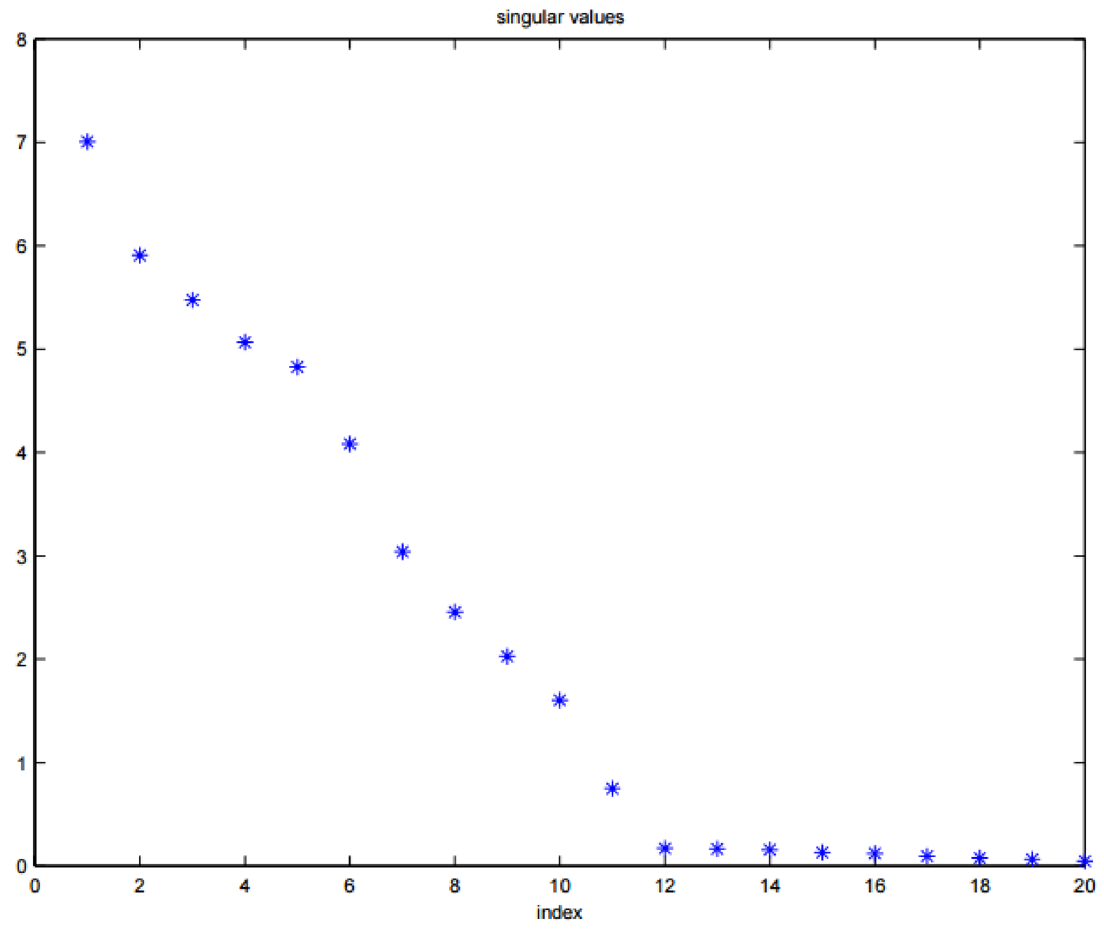
SVD Application:
Removing Noise/Dimensionality Reduction

Removing Noise Using SVD

- Assume that A contains some data matrix plus noise: A = A0 + N, where the noise N is small compared with A0.
- Then typically the singular values of A have the behavior illustrated in this figure.
- We can remove the noise by truncating the singular value expansion.



Matrix Approximation Using SVD

- We can write SVD of A = U ∑ V ^{T} in its equivalent outer product form as:

$$A = \sigma_1 U_1 V_1^T + \cdots + \sigma_n U_n V_n^T$$

- The truncated SVD is very important, not only for removing noise but also for compressing data and approximating a given matrix by one of lower rank.
- The difference (L2 norm) between the original and approximation is given by (k+1)th singular values.

$$\mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T \qquad \|\mathbf{A} - \mathbf{A}_k\|_2 = \sigma_{k+1}$$

Example

- Use the first 2 singular values of matrix A to find an approximation for A. Determine the error between A and A_2.

```
> A
     [,1] [,2] [,3] [,4]
[1,]   16    2    3   13
[2,]    5   11   10    8
[3,]    9    7    6   12
[4,]    4   14   15    1
```

Solution in Python

```
A = np.array([[16,2,3,13],\
              [5,11,10,8],\
              [9,7,6,12],
              [4,14,15,1]])
print(A)
```

```
[[16  2  3 13]
 [ 5 11 10  8]
 [ 9  7  6 12]
 [ 4 14 15  1]]
```

```
u,s,vh = la.svd(A)
print(s)
```

```
[3.40000000e+01 1.78885438e+01 4.47213595e+00 1.08292355e-16]
```

```
A2 = u[:,:2]@np.diag(s)[:2,:2]@vh[:2,:]
print(A2)
```

```
[[14.5  2.5  2.5 14.5]
 [ 6.5 10.5 10.5  6.5]
 [10.5  6.5  6.5 10.5]
 [ 2.5 14.5 14.5  2.5]]
```

```
print(la.norm(A-A2))
```

```
4.47213595499958
```

Note that error matches the 3rd singular value.