

```
In [2]: import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

The MNIST dataset

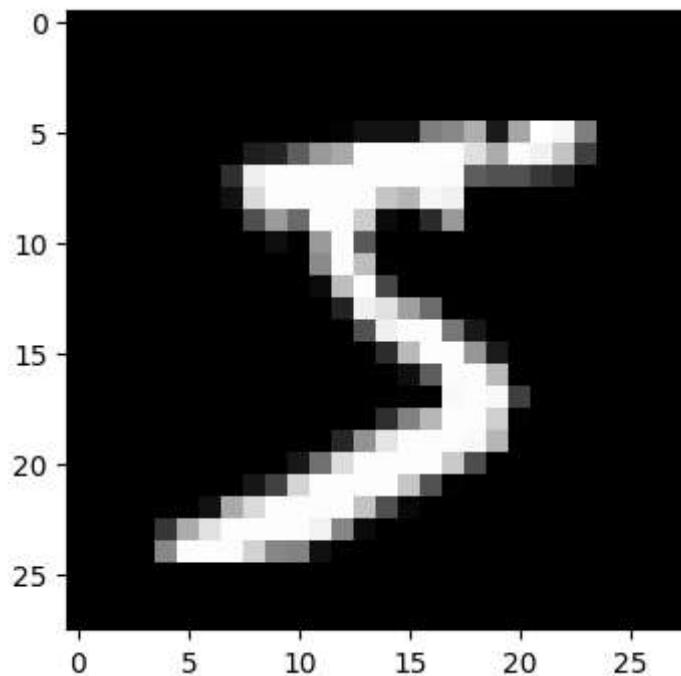
The MNIST dataset is composed of 70,000 28×28 grayscale images of handwritten digits. It is represented as a $70000 \times 28 \times 28$ numpy array (a "3d matrix").

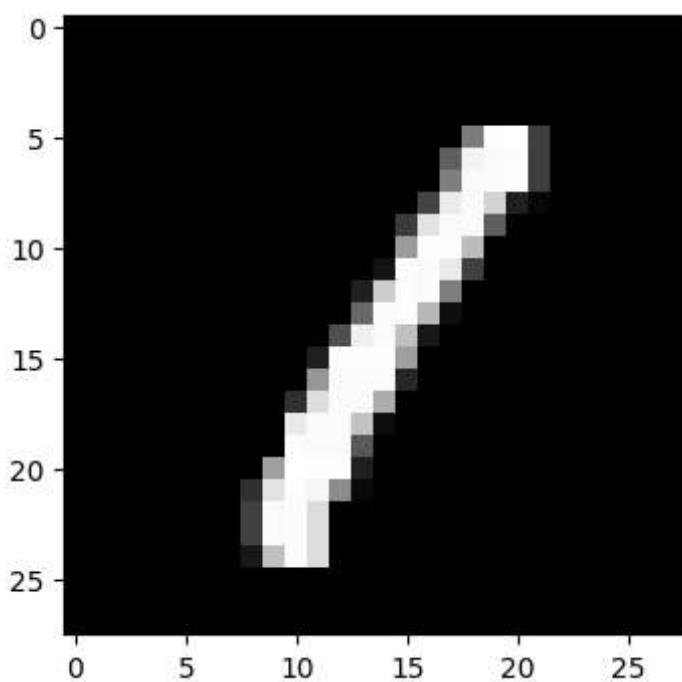
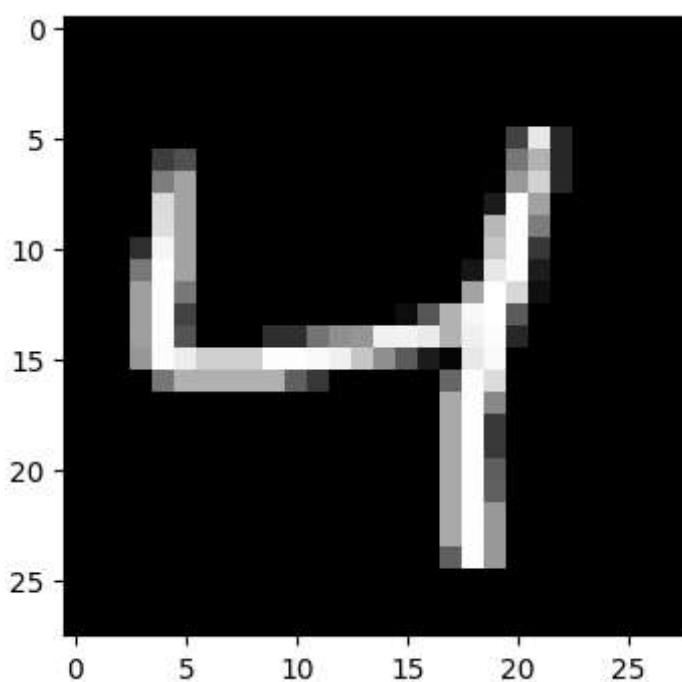
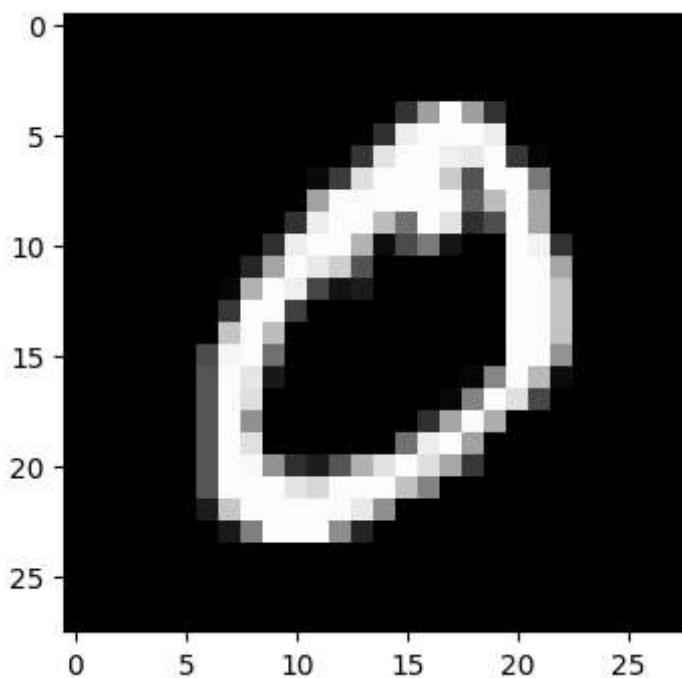
```
In [3]: x = np.load("mnist.npy")
print(x.shape)
```

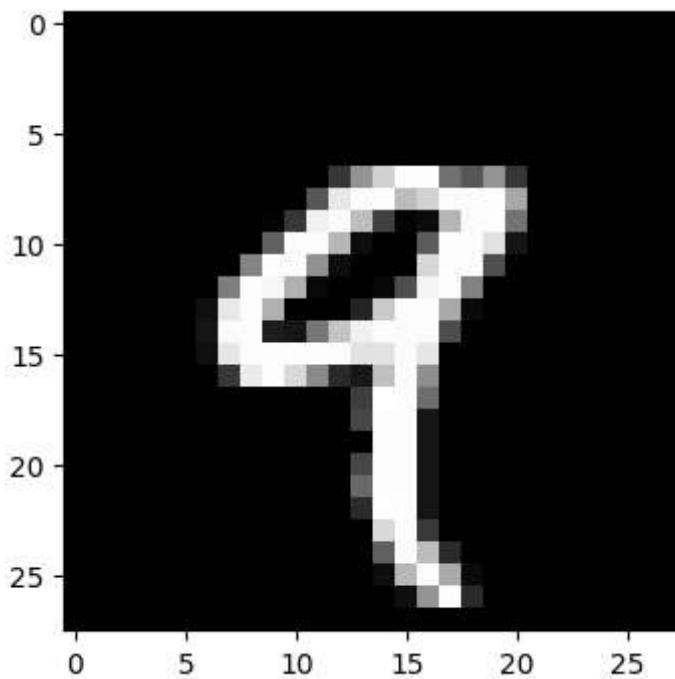
```
(70000, 28, 28)
```

Display the first few digits in the dataset.

```
In [5]: for i in range(5):
    plt.figure(figsize=(4,4))
    plt.imshow(x[i], cmap="gray")
    plt.show()
```







Computing and diagonalizing the covariance of MNIST

We will interpret each image as a vector in \mathbb{R}^d with $d = 28^2 = 768$. The dataset can thus be seen as a matrix in $\mathbb{R}^{n \times d}$ where $n = 70000$.

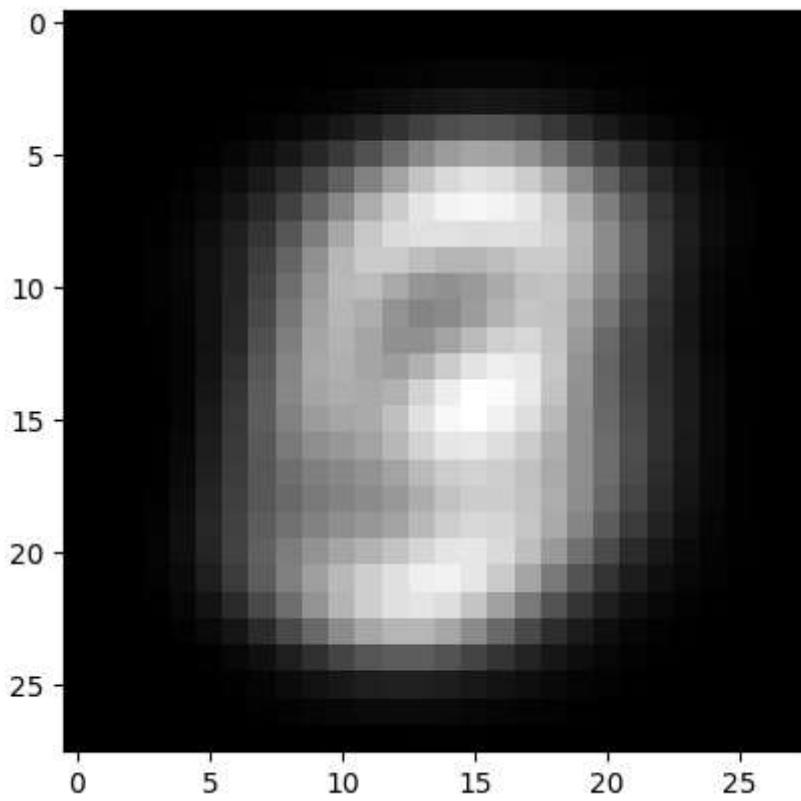
```
In [8]: xx = x.reshape((x.shape[0], -1))  
xx
```

```
Out[8]: array([[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]], dtype=uint8)
```

1. Compute the mean $\mu \in \mathbb{R}^d$ of the MNIST dataset and plot it as a 28×28 image.

```
In [9]: # Your answer here  
mean_vector = xx.mean(axis=0)  
mean_image = mean_vector.reshape((28,28))  
plt.imshow(mean_image, cmap="gray")
```

```
Out[9]: <matplotlib.image.AxesImage at 0x1e24bbc96c0>
```



2. Compute the covariance $\Sigma \in \mathbb{R}^{d \times d}$ of the MNIST dataset and diagonalize it using the function
`np.linalg.eigh`.

```
In [10]: # Your answer here
cov_matrix = (1 / xx.shape[0]) * (xx - mean_vector).T @ (xx - mean_vector)
eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
eigenvalues, eigenvectors
```

```
Out[10]: (array([-5.15746893e-11, -3.68566679e-11, -2.92211626e-11, -2.79618506e-11,
-2.53277348e-11, -1.31927665e-11, -1.09494873e-11, -5.24653862e-12,
-4.97795591e-12, -2.78329396e-12, -1.99441793e-12, -1.48201503e-12,
-1.24654450e-12, -7.01845945e-13, -5.40819685e-13, -3.64230139e-13,
-3.54149030e-13, -2.97937985e-14, -1.08470422e-14, -8.96341966e-16,
-6.05659620e-16, -3.56535501e-16, -1.60072184e-16, -3.84226201e-17,
-1.57789003e-17, -9.51191829e-18, -3.31040598e-18, -1.99638673e-27,
-8.06288238e-28, -2.10334352e-28, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 1.80550928e-28, 6.37374188e-28,
1.78773343e-27, 2.70679504e-17, 3.23563278e-17, 1.60320976e-16,
3.69459723e-16, 2.06831993e-15, 2.74126308e-15, 1.00581385e-14,
5.63787867e-14, 1.78650829e-13, 3.16973949e-13, 1.36336816e-12,
2.09379140e-12, 2.27951665e-12, 3.68383755e-12, 4.01365961e-12,
4.63801908e-12, 5.16185921e-12, 1.56957074e-11, 2.21639321e-11,
2.55927918e-11, 2.69699372e-11, 5.18250829e-11, 3.63783622e-04,
7.12275304e-04, 7.46187435e-04, 2.04922089e-03, 2.14456232e-03,
2.76607718e-03, 5.90291543e-03, 1.10612190e-02, 1.42506462e-02,
1.44691497e-02, 1.45984210e-02, 1.60990338e-02, 2.03670506e-02,
2.05651368e-02, 2.33995668e-02, 2.56352614e-02, 3.03277624e-02,
3.52897625e-02, 3.72051706e-02, 4.89868742e-02, 4.90825730e-02,
5.22772447e-02, 6.06075811e-02, 7.13973480e-02, 8.81714878e-02,
9.22778769e-02, 9.53037760e-02, 1.12590500e-01, 1.30987655e-01,
1.31155251e-01, 1.42951649e-01, 1.63949445e-01, 1.65322602e-01,
2.01375676e-01, 2.19577418e-01, 2.47908273e-01, 2.67890493e-01,
2.76054308e-01, 2.85737324e-01, 3.14965594e-01, 3.78101465e-01,
3.87861890e-01, 4.00656451e-01, 4.02035706e-01, 4.03367698e-01,
4.18016975e-01, 4.65118498e-01, 5.00117687e-01, 5.35003860e-01,
5.37353584e-01, 6.03590123e-01, 6.31376733e-01, 6.33139934e-01,
6.49818481e-01, 6.97550053e-01, 7.20866069e-01, 8.22273131e-01,
8.88821174e-01, 9.09468572e-01, 9.25560023e-01, 9.31552373e-01,
9.36494545e-01, 9.67702032e-01, 9.99072374e-01, 1.17519144e+00,
1.25648533e+00, 1.28378552e+00, 1.29958881e+00, 1.31415847e+00,
1.43994637e+00, 1.53556413e+00, 1.53713721e+00, 1.63903449e+00,
1.74648652e+00, 1.78005293e+00, 1.88189642e+00, 1.89668496e+00,
1.95838882e+00, 2.06198014e+00, 2.08357066e+00, 2.08689895e+00,
2.14532164e+00, 2.20764876e+00, 2.28267628e+00, 2.38761241e+00,
2.52759098e+00, 2.71596869e+00, 2.78323308e+00, 2.82829633e+00,
2.87458792e+00, 2.87650083e+00, 3.17136512e+00, 3.21587871e+00,
3.39316742e+00, 3.50345740e+00, 3.56641194e+00, 3.63459200e+00,
3.69709862e+00, 3.71495430e+00, 3.74603233e+00, 3.93607147e+00,
4.13002814e+00, 4.37012607e+00, 4.41304556e+00, 4.46322849e+00,
4.48437339e+00, 5.12024636e+00, 5.33847426e+00, 5.46234076e+00,
5.54843130e+00, 5.61011135e+00, 5.74549793e+00, 5.97002408e+00,
5.98773174e+00, 6.14606867e+00, 6.24202035e+00, 6.57514051e+00,
6.76891461e+00, 6.90261200e+00, 6.97801398e+00, 7.20581244e+00,
7.24667407e+00, 7.37725498e+00, 7.39216402e+00, 7.91055099e+00,
8.16586976e+00, 8.58133458e+00, 8.64381246e+00, 9.44358006e+00,
9.90483311e+00, 9.96143826e+00, 1.01339318e+01, 1.04774013e+01,
1.05415661e+01, 1.06808205e+01, 1.07963229e+01, 1.08797766e+01,
1.09858588e+01, 1.13897514e+01, 1.16219620e+01, 1.30534384e+01,
1.31659894e+01, 1.33201106e+01, 1.35578435e+01, 1.40790940e+01,
1.42210000e+01, 1.49832980e+01, 1.52456053e+01, 1.53152183e+01,
1.54335263e+01, 1.55947897e+01, 1.64369554e+01, 1.65791916e+01,
1.66224773e+01, 1.67977224e+01, 1.78581884e+01, 1.82420096e+01,
1.84270935e+01, 1.88059060e+01, 1.88956381e+01, 1.94317426e+01,
1.99962036e+01, 2.11501252e+01, 2.14543153e+01, 2.17017039e+01,
2.30856540e+01, 2.31483719e+01, 2.40597028e+01, 2.44729927e+01,
2.46568082e+01, 2.50168679e+01, 2.54385864e+01, 2.58127039e+01,
2.60990654e+01, 2.66460495e+01, 2.66982748e+01, 2.68316431e+01,
2.69700275e+01, 2.76446305e+01, 2.89708991e+01, 2.97066652e+01,
```

3.06863178e+01,	3.12123594e+01,	3.13616240e+01,	3.15519695e+01,
3.20876359e+01,	3.43775175e+01,	3.52204728e+01,	3.57310886e+01,
3.61695675e+01,	3.70825181e+01,	3.76805842e+01,	3.84538934e+01,
3.88215397e+01,	3.92095193e+01,	3.93908376e+01,	3.97671565e+01,
4.09598624e+01,	4.16672338e+01,	4.19648973e+01,	4.20648248e+01,
4.30674937e+01,	4.41539653e+01,	4.55843683e+01,	4.60684224e+01,
4.63778232e+01,	4.67652707e+01,	4.77642974e+01,	4.85153424e+01,
4.94712210e+01,	5.03193380e+01,	5.09577214e+01,	5.22793234e+01,
5.27238619e+01,	5.28380735e+01,	5.40579781e+01,	5.42605636e+01,
5.52413625e+01,	5.55673649e+01,	5.87251000e+01,	5.92273181e+01,
6.16074837e+01,	6.18736748e+01,	6.30478814e+01,	6.44571381e+01,
6.52586720e+01,	6.61837532e+01,	6.68436045e+01,	6.77371795e+01,
6.82537663e+01,	6.95607191e+01,	7.09154612e+01,	7.15468097e+01,
7.27699797e+01,	7.33600531e+01,	7.39170039e+01,	7.60184571e+01,
7.69564605e+01,	7.92245732e+01,	8.01298562e+01,	8.04437649e+01,
8.22556021e+01,	8.27938637e+01,	8.34869956e+01,	8.39056738e+01,
8.46021468e+01,	8.56216126e+01,	8.64845358e+01,	8.79758016e+01,
9.17572026e+01,	9.32030974e+01,	9.43014453e+01,	9.50060204e+01,
9.71209273e+01,	9.91961460e+01,	9.94262526e+01,	1.00225284e+02,
1.01859042e+02,	1.02402432e+02,	1.05669829e+02,	1.07670531e+02,
1.09012621e+02,	1.09899832e+02,	1.11943287e+02,	1.13103692e+02,
1.17482523e+02,	1.17808266e+02,	1.18283388e+02,	1.19893875e+02,
1.20351973e+02,	1.21027623e+02,	1.24955241e+02,	1.26519410e+02,
1.27616201e+02,	1.29315476e+02,	1.30321805e+02,	1.31054312e+02,
1.32051774e+02,	1.37854228e+02,	1.38882127e+02,	1.40085999e+02,
1.41342275e+02,	1.41390309e+02,	1.43860266e+02,	1.44276612e+02,
1.45452414e+02,	1.48940834e+02,	1.52987583e+02,	1.55146133e+02,
1.58679578e+02,	1.58954773e+02,	1.60951137e+02,	1.62903028e+02,
1.65551034e+02,	1.68028123e+02,	1.69374075e+02,	1.70613498e+02,
1.73732931e+02,	1.75438990e+02,	1.77104929e+02,	1.79286450e+02,
1.81040456e+02,	1.81542837e+02,	1.86067901e+02,	1.86472177e+02,
1.86865321e+02,	1.90003975e+02,	1.91973940e+02,	1.93532802e+02,
1.96624573e+02,	1.99052842e+02,	2.00810672e+02,	2.02093042e+02,
2.04928960e+02,	2.05979180e+02,	2.09336803e+02,	2.11869824e+02,
2.16555534e+02,	2.17257527e+02,	2.19508299e+02,	2.22130239e+02,
2.26039389e+02,	2.29394732e+02,	2.31214679e+02,	2.33179982e+02,
2.37698261e+02,	2.38705315e+02,	2.41018524e+02,	2.44430231e+02,
2.46813352e+02,	2.48337057e+02,	2.50411529e+02,	2.51498579e+02,
2.53397869e+02,	2.55423498e+02,	2.60328743e+02,	2.61497004e+02,
2.64659787e+02,	2.67119277e+02,	2.68311644e+02,	2.71609741e+02,
2.72242164e+02,	2.77273654e+02,	2.77490743e+02,	2.81326090e+02,
2.82583576e+02,	2.83506597e+02,	2.86779427e+02,	2.90404218e+02,
2.90937880e+02,	2.93704639e+02,	2.95997538e+02,	2.98295456e+02,
3.01713616e+02,	3.04798583e+02,	3.05858964e+02,	3.07950605e+02,
3.10004063e+02,	3.12705557e+02,	3.14199378e+02,	3.16694745e+02,
3.20040657e+02,	3.21973886e+02,	3.22782763e+02,	3.25605296e+02,
3.32317924e+02,	3.34445296e+02,	3.37259147e+02,	3.40386589e+02,
3.40820358e+02,	3.43479745e+02,	3.44462254e+02,	3.48787168e+02,
3.52049597e+02,	3.53870119e+02,	3.59596527e+02,	3.64247461e+02,
3.64482428e+02,	3.65417042e+02,	3.68556893e+02,	3.71262777e+02,
3.74318992e+02,	3.76412431e+02,	3.77385401e+02,	3.77882192e+02,
3.81074952e+02,	3.84739774e+02,	3.91967624e+02,	3.93879632e+02,
3.97466521e+02,	3.98007092e+02,	4.00874220e+02,	4.02270808e+02,
4.07577409e+02,	4.08429588e+02,	4.10744735e+02,	4.13599210e+02,
4.16607163e+02,	4.18234250e+02,	4.22587757e+02,	4.27469699e+02,
4.32043336e+02,	4.32667776e+02,	4.34788875e+02,	4.36770228e+02,
4.37991259e+02,	4.41481917e+02,	4.44086976e+02,	4.48804662e+02,
4.54243818e+02,	4.58053123e+02,	4.60861885e+02,	4.63277446e+02,
4.65499878e+02,	4.67354617e+02,	4.70975261e+02,	4.72505950e+02,
4.81081806e+02,	4.85228354e+02,	4.87348062e+02,	4.89700216e+02,
4.91240827e+02,	4.95718588e+02,	5.01117393e+02,	5.06861478e+02,
5.07537622e+02,	5.12685943e+02,	5.14221583e+02,	5.20363220e+02,
5.24924669e+02,	5.26821174e+02,	5.29977729e+02,	5.35087157e+02,

5.39822006e+02,	5.44811173e+02,	5.47435656e+02,	5.48735226e+02,
5.54705033e+02,	5.58818681e+02,	5.60742207e+02,	5.62477074e+02,
5.70435639e+02,	5.71380982e+02,	5.72948647e+02,	5.78761154e+02,
5.83572144e+02,	5.87305158e+02,	5.90713381e+02,	5.93652473e+02,
5.96059660e+02,	6.02224202e+02,	6.05303005e+02,	6.06981068e+02,
6.17310073e+02,	6.24228721e+02,	6.25182970e+02,	6.30717515e+02,
6.34561764e+02,	6.36516649e+02,	6.42145817e+02,	6.42404979e+02,
6.47141676e+02,	6.54534665e+02,	6.57712430e+02,	6.60728481e+02,
6.61571514e+02,	6.70585856e+02,	6.77359436e+02,	6.80593397e+02,
6.88769684e+02,	6.90502127e+02,	6.95427061e+02,	7.01867401e+02,
7.07662521e+02,	7.18103611e+02,	7.19887572e+02,	7.29076370e+02,
7.30495864e+02,	7.35718506e+02,	7.41286689e+02,	7.45578233e+02,
7.53773383e+02,	7.57633005e+02,	7.65514695e+02,	7.78804491e+02,
7.80318161e+02,	7.87104531e+02,	7.92842319e+02,	7.94858549e+02,
8.11070741e+02,	8.17599434e+02,	8.20305078e+02,	8.25694664e+02,
8.26831843e+02,	8.31965586e+02,	8.40828284e+02,	8.46789600e+02,
8.52323278e+02,	8.63436933e+02,	8.66930513e+02,	8.72771453e+02,
8.79973635e+02,	8.83027334e+02,	8.87256218e+02,	8.97170588e+02,
9.04254026e+02,	9.07520813e+02,	9.15631354e+02,	9.20967488e+02,
9.33125949e+02,	9.41003264e+02,	9.51358279e+02,	9.65752682e+02,
9.70440032e+02,	9.83108138e+02,	1.00070169e+03,	1.00443355e+03,
1.01225187e+03,	1.01361479e+03,	1.03008099e+03,	1.03318537e+03,
1.03772616e+03,	1.04909379e+03,	1.06115846e+03,	1.06283550e+03,
1.08165066e+03,	1.08598349e+03,	1.09812628e+03,	1.10716183e+03,
1.11190082e+03,	1.12459323e+03,	1.12884286e+03,	1.14886983e+03,
1.16232820e+03,	1.17254755e+03,	1.18261772e+03,	1.18889238e+03,
1.20835843e+03,	1.21334569e+03,	1.22392252e+03,	1.24176813e+03,
1.25125972e+03,	1.25647484e+03,	1.27616808e+03,	1.29118792e+03,
1.29819912e+03,	1.31504512e+03,	1.33467575e+03,	1.34371197e+03,
1.35138111e+03,	1.36600989e+03,	1.40085321e+03,	1.41658533e+03,
1.43432829e+03,	1.44352977e+03,	1.46245486e+03,	1.50232345e+03,
1.51229177e+03,	1.53379021e+03,	1.54178420e+03,	1.57190394e+03,
1.58462080e+03,	1.59229687e+03,	1.60081645e+03,	1.61746831e+03,
1.64947486e+03,	1.65936033e+03,	1.68872692e+03,	1.70074976e+03,
1.70861096e+03,	1.72285944e+03,	1.74895725e+03,	1.79298228e+03,
1.80322325e+03,	1.83700715e+03,	1.89436314e+03,	1.93237154e+03,
1.96450855e+03,	1.98562462e+03,	2.00121802e+03,	2.01092162e+03,
2.03670133e+03,	2.05888707e+03,	2.06279873e+03,	2.12529973e+03,
2.15541207e+03,	2.17039681e+03,	2.20434235e+03,	2.26814076e+03,
2.30406565e+03,	2.33025185e+03,	2.35611760e+03,	2.37267064e+03,
2.41266139e+03,	2.45530218e+03,	2.48797873e+03,	2.50470790e+03,
2.56831503e+03,	2.60873990e+03,	2.62320064e+03,	2.66101051e+03,
2.66681791e+03,	2.69604170e+03,	2.80306509e+03,	2.88460967e+03,
2.92421912e+03,	2.95520480e+03,	3.05096293e+03,	3.08951630e+03,
3.12698352e+03,	3.20986784e+03,	3.22970705e+03,	3.35017949e+03,
3.42765970e+03,	3.45023404e+03,	3.54722356e+03,	3.55661420e+03,
3.67608725e+03,	3.71614290e+03,	3.76822865e+03,	3.86033046e+03,
3.92110985e+03,	3.99121859e+03,	4.13034968e+03,	4.20273042e+03,
4.29336843e+03,	4.43143921e+03,	4.52389433e+03,	4.54030783e+03,
4.63130588e+03,	4.78841137e+03,	4.80698011e+03,	4.83681911e+03,
4.87740585e+03,	5.04276386e+03,	5.29258958e+03,	5.53625504e+03,
5.59898869e+03,	5.65429098e+03,	5.94615741e+03,	6.06375054e+03,
6.17752986e+03,	6.41323378e+03,	6.46719983e+03,	6.56739370e+03,
6.68965480e+03,	6.95534937e+03,	7.07663633e+03,	7.30791983e+03,
7.57849385e+03,	7.86265900e+03,	8.18326738e+03,	8.22172548e+03,
8.39136490e+03,	8.68359267e+03,	8.79820473e+03,	9.20578748e+03,
9.24113022e+03,	9.69015100e+03,	9.84750631e+03,	1.01692003e+04,
1.06378912e+04,	1.08680122e+04,	1.09654969e+04,	1.16184074e+04,
1.19714546e+04,	1.23968292e+04,	1.28893227e+04,	1.31610777e+04,
1.35883208e+04,	1.43447224e+04,	1.52605741e+04,	1.55847279e+04,
1.60383427e+04,	1.64280388e+04,	1.67066997e+04,	1.73116665e+04,
1.86409047e+04,	1.94395023e+04,	2.00863333e+04,	2.06079485e+04,
2.21394661e+04,	2.25055712e+04,	2.36673094e+04,	2.53908053e+04,

```

2.69499231e+04, 2.77770236e+04, 2.87712502e+04, 3.02965122e+04,
3.12002508e+04, 3.28986421e+04, 3.46356878e+04, 3.65648922e+04,
3.95454343e+04, 4.07231430e+04, 4.38698621e+04, 4.52536592e+04,
5.09812503e+04, 5.43096063e+04, 5.81044457e+04, 5.85518694e+04,
6.98875556e+04, 7.22588790e+04, 8.03348093e+04, 9.46111872e+04,
9.91139674e+04, 1.12443533e+05, 1.47668187e+05, 1.67689177e+05,
1.85334709e+05, 2.10927341e+05, 2.45429921e+05, 3.34289286e+05]),
array([[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.])))

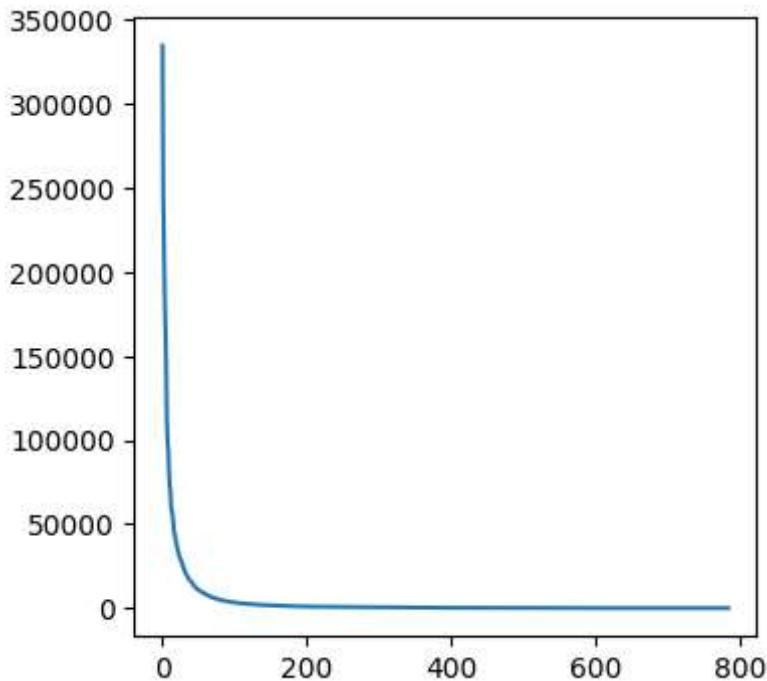
```

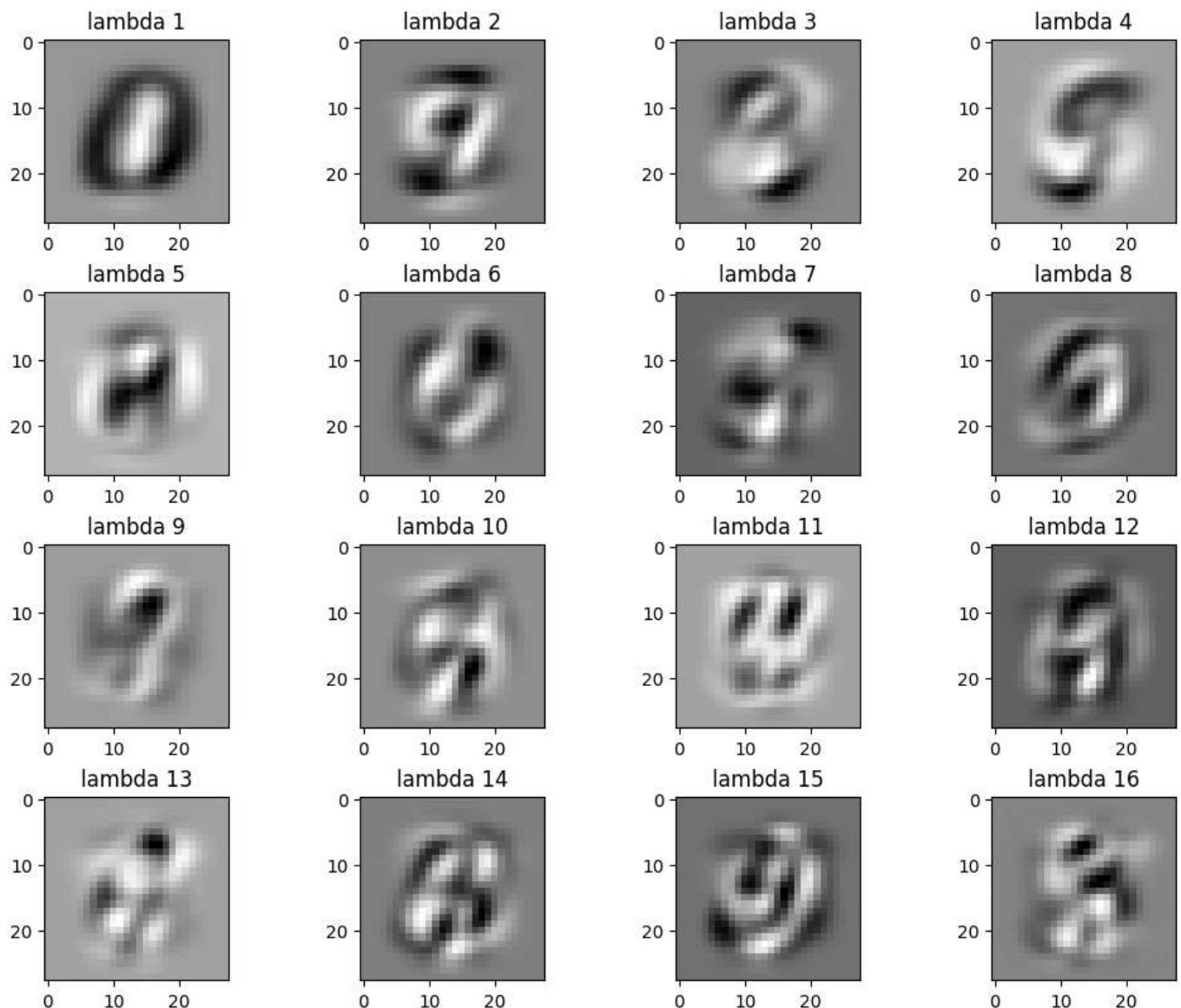
3. Plot the ordered eigenvalues $\lambda_1 \geq \dots \geq \lambda_k \geq \dots$ as a function $k = 1, \dots, d$ with the x axis in log scale, and the first few eigenvectors u_1, \dots, u_k, \dots as 28×28 images.

```
In [12]: # Your answer here
sorted_eigenvalues = sorted(eigenvalues, reverse=True)
sorted_eigenvectors = np.fliplr(eigenvectors)

plt.figure(figsize=(4,4))
plt.plot(range(1,len(sorted_eigenvalues)+1),sorted_eigenvalues)
fig, axes = plt.subplots(4,4, layout='constrained', figsize=(10, 8))
for i in range(4):
    for j in range(4):
        index = 4 * i + j
        axes[i][j].imshow(sorted_eigenvectors[:,index].reshape(28, 28), cmap="gray")
        axes[i][j].set_title(f"lambda {index + 1}")

# Your answer here
```





PCA compression of MNIST

4. Let $k \in \mathbb{N}$. Compute the k -dimensional PCA approximation z_1, \dots, z_n of the MNIST dataset using the eigenvectors u_1, \dots, u_k . Then, compute the reconstructed images $\hat{x}_i = \mu + z_{i,1}u_1 + \dots + z_{i,k}u_k$, which are equal to the mean μ plus the orthogonal projection of $x_i - \mu$ on $\text{Span}(u_1, \dots, u_k)$. Display the first 5 reconstructed images $\hat{x}_1, \dots, \hat{x}_5$. Choose a small value of k that still allows recognizing the digits.

```
In [10]: # Your answer here
# Compute the dimension of each data point
def dim_data(data):
    return data.shape[1]

# Compute the mean of the sample
def mean_data(data):
    return np.mean(data, axis=0)

# Compute the standard deviation of the sample
def sd_data(data):
    return (data - np.mean(data, axis=0)) / np.sqrt(np.sum((data - np.mean(data, axis=0))**2, axis=0))

# Centerize the data for further computation of the covariance matrix
def center_data(data):
    return data - np.mean(data, axis=0)
```

```

def centerize_data(data):
    # Centerize the data
    return data - np.mean(data, axis=0)

# Compute the eigenbasis with k eigenvectors
def compute_eigenbasis_k(centered_data, k):
    # Compute the top k eigenvectors for our eigenbasis
    cov_matrix = 1 / len(centered_data) * centered_data.T @ centered_data

    eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)

    # Flip the columns, in reversed order.
    sorted_eigenvectors = np.fliplr(eigenvectors)[:, :k]

    return sorted_eigenvectors

# Main Procedure: PCA
def PCA_procedure(data, k):
    centered_data = centerize_data(data)
    V_k = compute_eigenbasis_k(centered_data, k)

    # Find PCA coordinates
    Z_k = V_k.T @ centered_data.T

    # Z_k is a matrix with (k, 70000), where the coordinates for each data point onto eigenba
    return Z_k, V_k

# Main Procesure: Inverse PCA
def inverse_PCA_procedure(Z_k, V_k, data, k):
    centered_data = centerize_data(data)

    mu = data.mean(axis=0)

    # Revert to origin
    RC_k = V_k @ Z_k + mu.reshape(dim_data(data), 1)

    # RC_k is a matrix with (784, 70000), where the reconstructed coordinates for each data p
    return RC_k

# Display the reconstructed images
def show_first_five_reconstructed(data, k):
    # Draw the first five reconstructed images
    fig, axes = plt.subplots(1, 5, figsize=(16, 16), constrained_layout=True)

    Z_k, V_k = PCA_procedure(data, k)
    RC_k = inverse_PCA_procedure(Z_k, V_k, data, k)

    for i in range(5):
        axes[i].imshow(RC_k[:, i].reshape(28, 28), cmap="gray")

    plt.title(f"Reconstructed images when k = {k}", loc="left")

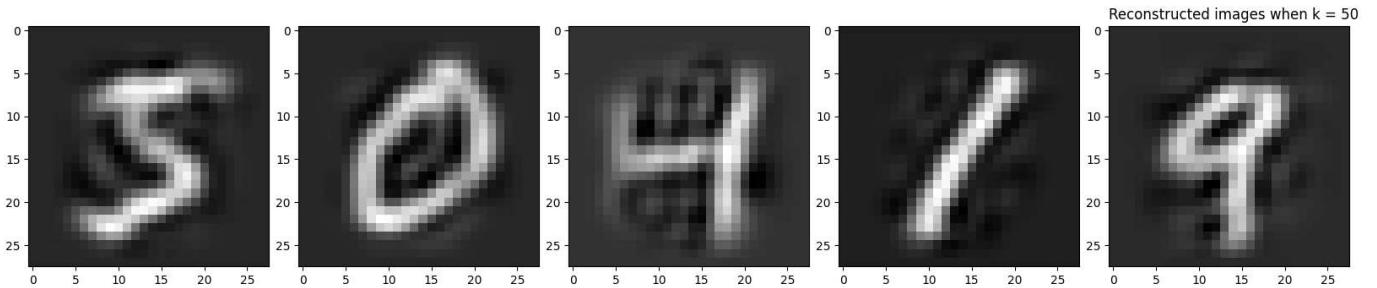
def problem_1(data):
    k = 50
    show_first_five_reconstructed(data, k)

def problem_2(data):

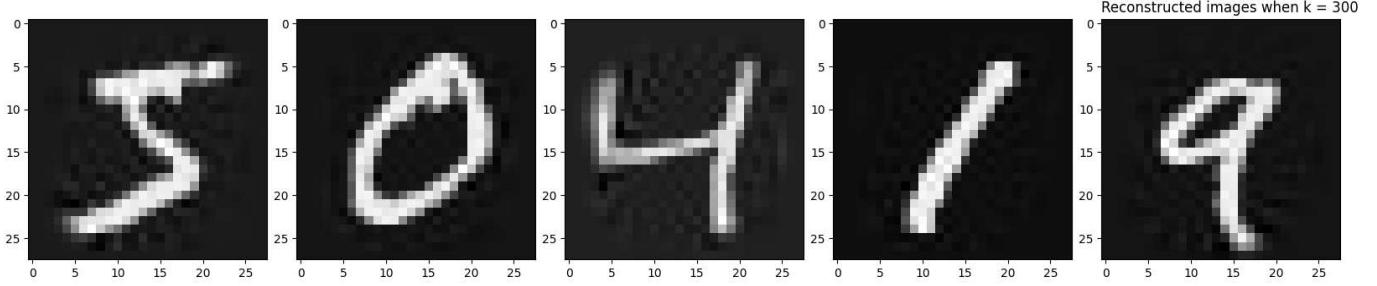
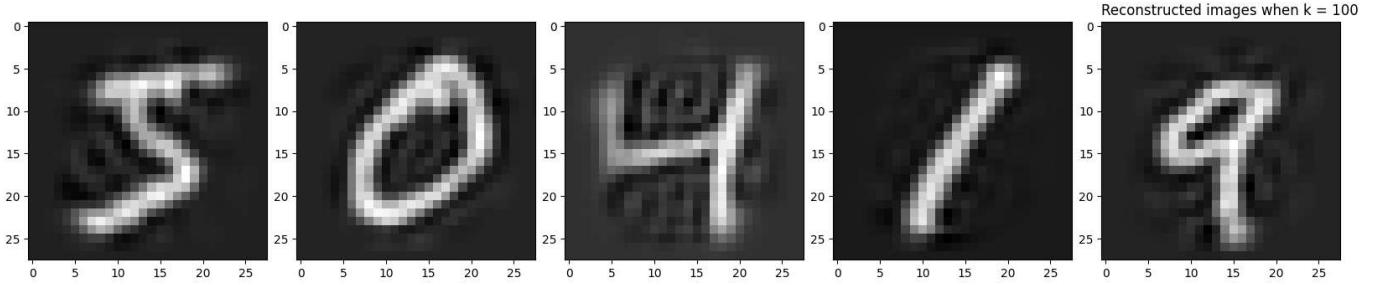
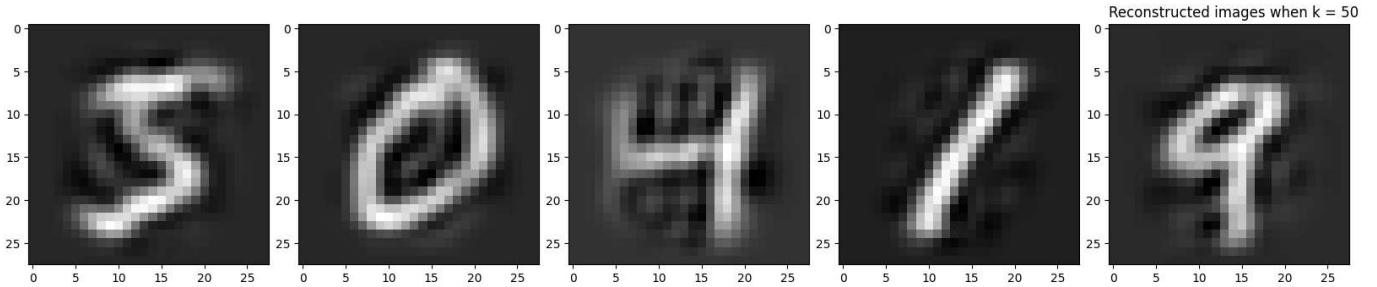
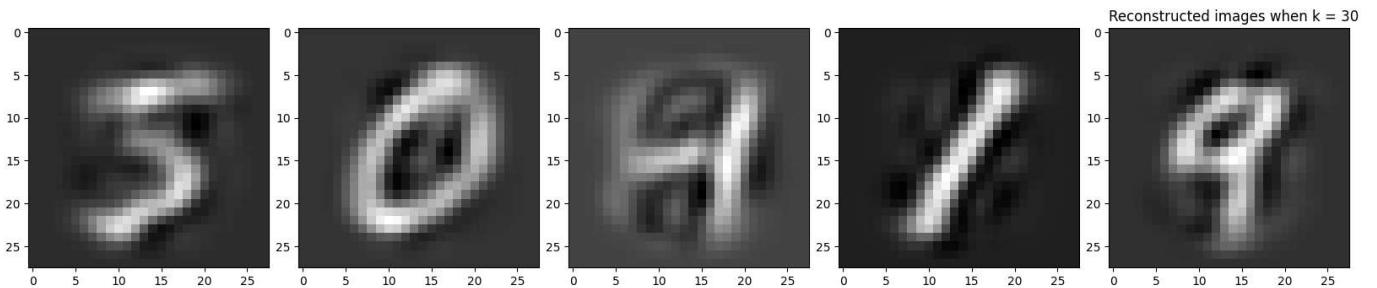
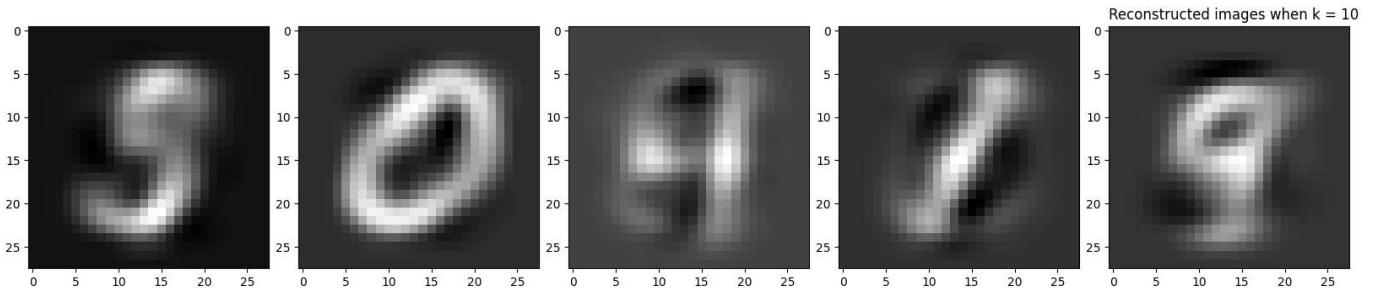
```

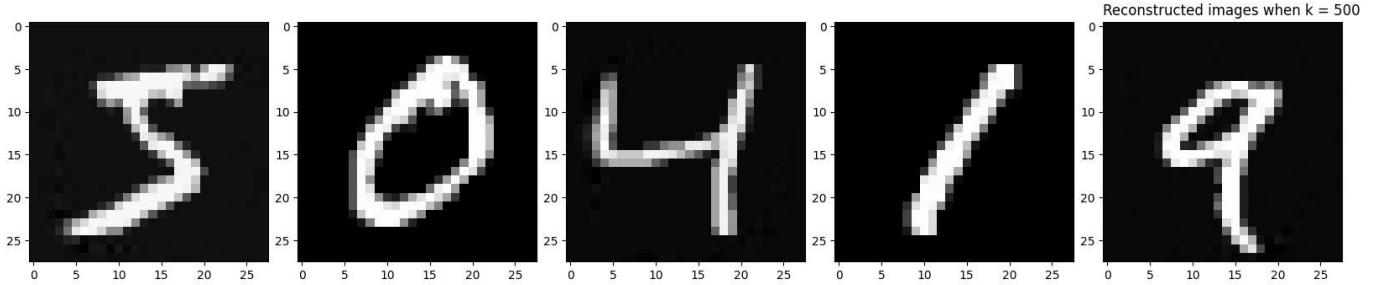
```
k_list = [10,30,50,100,300,500]
for k in k_list:
    show_first_five_reconstructed(data, k)
```

```
# xx is (70000, 784)
problem_1(xx) # We pick 50, when we could recognize the reconstructed digits by raw eyes, whi
```



```
In [11]: problem_2(xx) # We find that 50 is minimum number of eigenvectors that are needed to reconstr
```





In []: