Linear Algebra
Homework 4: Factorization and iteration

Solutions are by Yaroslava Lochman.

Problem 1 (Rank of a matrix; 2 pt)**.** (a) Assume that A is an $m \times n$ matrix of rank r. Without using the singular value decomposition, prove that A can be written as a sum of r summands $\mathbf{u}_j\mathbf{v}_j^\top$ of rank 1. Is such a representation unique?

(b) For the matrices below, find their ranks r and represent them as the sum of r rank-one summands (do not use SVD yet!).

$$A = \begin{pmatrix} 1 & -2 & 3 & -4 \\ -2 & 4 & -6 & 8 \\ 3 & -6 & 9 & -12 \\ -4 & 8 & -12 & 16 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{pmatrix}.$$

Solution to the problem 1. .

(a) Let $A = \begin{pmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{pmatrix}, \quad a_i \in \mathbb{R}^m \quad i = \overline{1,n}.$

$$\operatorname{rank} A = r \;\Rightarrow\; \exists\{\mathbf{u}_1, \dots, \mathbf{u}_r\} \text{ – basis of } \mathcal{C}(A),\ \mathbf{u}_j \in \mathbb{R}^m$$

$$\Rightarrow\; \forall i = \overline{1,n} \quad \exists\{v_1^{(i)}, \dots, v_r^{(i)}\},\ v_j^{(i)} \in \mathbb{R} : \quad \mathbf{a}_i = \sum_{j=1}^{r} v_j^{(i)}\mathbf{u}_j$$

$$\mathbf{v}_j := \begin{pmatrix} v_j^{(1)} & \cdots & v_j^{(n)} \end{pmatrix}^\top \;\Rightarrow\; A = \begin{pmatrix} \sum_{j=1}^r v_j^{(1)}\mathbf{u}_j & \cdots & \sum_{j=1}^r v_j^{(n)}\mathbf{u}_j \end{pmatrix} = \sum_{j=1}^{r} \mathbf{u}_j\mathbf{v}_j^\top$$

Since the basis of the vector space is not unique (can be constructed in different ways), the representation is not unique.

(b)

$$A = \begin{pmatrix} 1 & -2 & 3 & -4 \\ -2 & 4 & -6 & 8 \\ 3 & -6 & 9 & -12 \\ -4 & 8 & -12 & 16 \end{pmatrix} = \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_4 \end{pmatrix}$$

One can see that $\mathbf{a}_2 = -2\mathbf{a}_1,\ \mathbf{a}_3 = 3\mathbf{a}_1,\ \mathbf{a}_4 = -4\mathbf{a}_1$, so $\operatorname{rank} A = 1$ and:

$$\{\mathbf{u}_1\} = \{\mathbf{a}_1\} = \left\{ \begin{pmatrix} 1 \\ -2 \\ 3 \\ -4 \end{pmatrix} \right\} \text{ – basis of } \mathcal{C}(A), \qquad \{\mathbf{v}_1\} = \left\{ \begin{pmatrix} 1 \\ -2 \\ 3 \\ -4 \end{pmatrix} \right\} \quad \text{and} \quad A = \mathbf{u}_1\mathbf{v}_1^\top$$

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 & \mathbf{b}_4 \end{pmatrix}$$

Let $\mathbf{u} = \begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix}^\top$. One can see that $\mathbf{b}_2 = \mathbf{b}_1 + \mathbf{u}$, $\mathbf{b}_3 = \mathbf{b}_1 + 2\mathbf{u}$, $\mathbf{b}_4 = \mathbf{b}_1 + 3\mathbf{u}$, so $\operatorname{rank} B = 2$ and:

$$\{\mathbf{u}_1, \mathbf{u}_2\} = \{\mathbf{b}_1, \mathbf{u}\} = \left\{ \begin{pmatrix}1\\2\\3\\4\end{pmatrix}, \begin{pmatrix}1\\1\\1\\1\end{pmatrix} \right\} - \text{basis of } \mathcal{C}(B), \qquad \{\mathbf{v}_1, \mathbf{v}_2\} = \left\{ \begin{pmatrix}1\\1\\1\\1\end{pmatrix}, \begin{pmatrix}0\\1\\2\\3\end{pmatrix} \right\}$$

$$B = \mathbf{u}_1\mathbf{v}_1^\top + \mathbf{u}_2\mathbf{v}_2^\top$$

Answer: $A = \begin{pmatrix}1 & -2 & 3 & -4\end{pmatrix}^\top \begin{pmatrix}1 & -2 & 3 & -4\end{pmatrix}$

$$B = \begin{pmatrix}1 & 2 & 3 & 4\end{pmatrix}^\top \begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix} + \begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix}^\top \begin{pmatrix}0 & 1 & 2 & 3\end{pmatrix}$$

Problem 2 (Singular values; 3 pt)**.** (a) Prove that matrices A and $A^\top$ have the same non-zero singular values.

(b) Find singular values of the following matrices:

$$(i) \quad \begin{pmatrix}0 & 1 & 2\end{pmatrix}; \qquad (ii) \quad \begin{pmatrix}0 & 1 & 2\end{pmatrix}^\top; \qquad (iii) \quad \begin{pmatrix}1 & 2 & 2\\1 & 3 & 3\end{pmatrix}; \qquad (iv) \quad \begin{pmatrix}2 & 0\\1 & 2\end{pmatrix}.$$

Hint: Use (a) in (i) and (iii)

Solution to the problem 2. .

(a) Let A be $m \times n$ matrix.

$$A = U\Sigma V^\top - \text{ SVD of } A \;\Rightarrow\; A^\top = (U\Sigma V^\top)^\top = V\Sigma^\top U^\top$$

V and U are orthogonal. $\Sigma^\top$ is $n \times m$ and has non-zero values only on the main diagonal (because Σ has singular values of A on the main diagonal and zeros otherwise). Therefore $A^\top = V\Sigma^\top U^\top$ – SVD of $A^\top$. And since the main diagonal of $\Sigma^\top$ corresponds to singular values of $A^\top$, A and $A^\top$ have the same non-zero singular values.

(b) (i)

$$A = \begin{pmatrix}0 & 1 & 2\end{pmatrix}$$

Let's find singular values for $A^\top$:

$$(A^\top)^\top A^\top = AA^\top = \begin{pmatrix}5\end{pmatrix}$$

$$\Rightarrow \; \lambda_1 = 5 \quad \Rightarrow \quad \sigma_1 = \sqrt{5}$$

The same singular value $\sigma_1 = \sqrt{5}$ is for A.
Answer: $\sigma_1 = \sqrt{5}$

(ii)

$$A = \begin{pmatrix}0 & 1 & 2\end{pmatrix}^\top$$

From (i): $\sigma_1 = \sqrt{5}$.
Answer: $\sigma_1 = \sqrt{5}$

(iii)

$$A = \begin{pmatrix} 1 & 2 & 2 \\ 1 & 3 & 3 \end{pmatrix}$$

Let's find singular values for $A^\top$:

$$AA^\top = \begin{pmatrix} 9 & 13 \\ 13 & 19 \end{pmatrix}$$

$$\lambda^2 - 28\lambda + 2 = 0 \quad \Rightarrow \quad \lambda_{1,2} = 14 \pm \sqrt{194} \quad \Rightarrow \quad \sigma_{1,2} = \sqrt{14 \pm \sqrt{194}}$$

Answer: $\sigma_1 = \sqrt{14 + \sqrt{194}},\ \sigma_2 = \sqrt{14 - \sqrt{194}}$

(iv)

$$A = \begin{pmatrix} 2 & 0 \\ 1 & 2 \end{pmatrix} \quad A^\top A = \begin{pmatrix} 5 & 2 \\ 2 & 4 \end{pmatrix}$$

$$\lambda^2 - 9\lambda + 16 = 0 \quad \Rightarrow \quad \lambda_{1,2} = \frac{9 \pm \sqrt{17}}{2} \quad \Rightarrow \quad \sigma_{1,2} = \sqrt{\frac{9 \pm \sqrt{17}}{2}}$$

Answer: $\sigma_1 = \sqrt{\frac{9+\sqrt{17}}{2}},\ \sigma_2 = \sqrt{\frac{9-\sqrt{17}}{2}}$

Problem 3 (Singular value decomposition; 4 pt)**.** (a) If $A = U\Sigma V^\top$ is the SVD of A, what is the SVD of $A^\top$?

(b) Assume that $A = \mathbf{u}\mathbf{v}^\top$ is an $m \times n$ matrix of rank 1. Find the SVD of A.

(c) Find SVD of the following matrices:

$$(i) \quad \begin{pmatrix} 0 & 1 & 2 \end{pmatrix}; \qquad (ii) \quad \begin{pmatrix} 0 & 1 & 2 \end{pmatrix}^\top; \qquad (iii) \quad \begin{pmatrix} 2 & 1 & -2 \\ -2 & -1 & 2 \end{pmatrix}.$$

Solution to the problem 3. .

(a) From the **Problem 2** (a): $A^\top = V\Sigma^\top U^\top$. So let $\hat{A} = A^\top$, then $\hat{U} = V$; $\hat{\Sigma} = \Sigma^\top$; $\hat{V} = U$.

(b)

$$A = \mathbf{u}\mathbf{v}^\top = (\|\mathbf{u}\|\,\|\mathbf{v}\|)\frac{\mathbf{u}}{\|\mathbf{u}\|}\frac{\mathbf{v}}{\|\mathbf{v}\|}^\top = \sigma\hat{\mathbf{u}}\hat{\mathbf{v}}^\top$$

$$\sigma = \|\mathbf{u}\|\,\|\mathbf{v}\| \qquad \hat{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|} \qquad \hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

Answer: $A = \|\mathbf{u}\|\,\|\mathbf{v}\| \cdot \frac{\mathbf{u}}{\|\mathbf{u}\|} \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|}^\top$

(c) (i)

$$A = \begin{pmatrix} 0 & 1 & 2 \end{pmatrix}$$

First let's solve the next item (ii). So for $A^\top$: $\hat{\sigma}_1 = \sqrt{5} \quad \hat{\mathbf{v}}_1 = \begin{pmatrix} 1 \end{pmatrix} \quad \hat{\mathbf{u}}_1 = \frac{1}{\sqrt{5}}\begin{pmatrix} 0 & 1 & 2 \end{pmatrix}^\top$
Hence for A we will have: $\sigma_1 = \hat{\sigma}_1 = \sqrt{5} \quad \mathbf{v}_1 = \hat{\mathbf{u}}_1 = \frac{1}{\sqrt{5}}\begin{pmatrix} 0 & 1 & 2 \end{pmatrix}^\top \quad \mathbf{u}_1 = \hat{\mathbf{v}}_1 = \begin{pmatrix} 1 \end{pmatrix}$
Answer: $A = \sqrt{5} \cdot \begin{pmatrix} 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1/\sqrt{5} & 2/\sqrt{5} \end{pmatrix}$

(ii)

$$A = \begin{pmatrix} 0 & 1 & 2 \end{pmatrix}^\top \qquad \operatorname{rank} A = 1$$

$$A^\top A = \begin{pmatrix} 5 \end{pmatrix}$$

$$\lambda_1 = 5 \quad \sigma_1 = \sqrt{5} \qquad \mathbf{v}_1 = \begin{pmatrix} 1 \end{pmatrix} \qquad \mathbf{u}_1 = \frac{A\mathbf{v}_1}{\sigma_1} = \frac{1}{\sqrt{5}} \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$$

Answer: $A = \sqrt{5} \cdot \begin{pmatrix} 0 \\ 1/\sqrt{5} \\ 2/\sqrt{5} \end{pmatrix} \cdot \begin{pmatrix} 1 \end{pmatrix}$

(iii)

$$A = \begin{pmatrix} 2 & 1 & -2 \\ -2 & -1 & 2 \end{pmatrix} \qquad \operatorname{rank} A = 1 \quad - \quad \text{number of non-zero singular values of } A$$

Let's solve for $\hat{A} = A^\top$:

$$\hat{A} = \begin{pmatrix} 2 & -2 \\ 1 & -1 \\ -2 & 2 \end{pmatrix} \qquad C = \hat{A}^\top \hat{A} = \begin{pmatrix} 9 & -9 \\ -9 & 9 \end{pmatrix}$$

$$\lambda_1 = 18 \quad \lambda_2 = 0$$

$$\hat{\sigma}_1 = 3\sqrt{2} \quad \hat{\sigma}_2 = 0$$

$$(C - 18I)\mathbf{v} = 0 \;\Rightarrow\; \hat{\mathbf{v}}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$\hat{\mathbf{u}}_1 = \frac{1}{3\sqrt{2}} \frac{1}{\sqrt{2}} \begin{pmatrix} 2 & -2 \\ 1 & -1 \\ -2 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 4 \\ 2 \\ -4 \end{pmatrix}$$

Hence for A we have:

$$\sigma_1 = \hat{\sigma}_1 = 3\sqrt{2} \qquad \mathbf{v}_1 = \hat{\mathbf{u}}_1 = \frac{1}{6} \begin{pmatrix} 4 \\ 2 \\ -4 \end{pmatrix} \qquad \mathbf{u}_1 = \hat{\mathbf{v}}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Answer: $A = 3\sqrt{2} \cdot \begin{pmatrix} 2/3 \\ 1/3 \\ -2/3 \end{pmatrix} \cdot \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}$

Problem 4 (Singular value decomposition, 4 pt)**.** Find the singular value decomposition of the matrix

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

To this end, complete the following steps:

(a) How many singular values σ_j does A have? How many of them are non-zero? Find the nonzero singular values $\sigma_1, \ldots, \sigma_r$.

(b) find the right singular vectors $A\mathbf{v}_j = \sigma_j \mathbf{u}_j$, $j = 1, \ldots, r$;

(c) find the left singular vectors $A^\top \mathbf{u}_j = \sigma_j \mathbf{v}_j$, $j = 1, \ldots, r$;

(d) form the unitary matrices U and V and write the singular value decomposition of A.

(Hint: you may find it easier to work with $A^\top$)

Solution to the problem 4. .

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix} \quad - \quad 2 \times 3 \text{ matrix} \qquad \operatorname{rank} A = 2$$

(a) A has $\min\{2,\ 3\} = 2$ singular values σ_j, the number of non-zero singular values is $\operatorname{rank} A = 2$. To find it let's consider $\hat{A} = A^\top$:

$$\hat{A} = \begin{pmatrix} 3 & 2 \\ 2 & 3 \\ 2 & -2 \end{pmatrix} \qquad C = \hat{A}^\top \hat{A} = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix} \qquad \lambda_1 = 25 \quad \lambda_2 = 34 - 25 = 9$$

$$\Rightarrow \ \sigma_1 = 5 \quad \sigma_2 = 3$$

(b) and (c):

$$(C - 25I)\mathbf{v} = 0 \ \Rightarrow \ \hat{\mathbf{v}}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \hat{\mathbf{u}}_1 = \frac{\hat{A}\mathbf{v}_1}{\sigma_1} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

$$(C - 9I)\mathbf{v} = 0 \ \Rightarrow \ \hat{\mathbf{v}}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \qquad \hat{\mathbf{u}}_2 = \frac{\hat{A}\mathbf{v}_2}{\sigma_1} = \frac{1}{3\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ 4 \end{pmatrix}$$

Hence for A:

$$\mathbf{v}_1 = \hat{\mathbf{u}}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \qquad \mathbf{v}_2 = \hat{\mathbf{u}}_2 = \frac{1}{3\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ 4 \end{pmatrix}$$

$$\mathbf{u}_1 = \hat{\mathbf{v}}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \mathbf{u}_2 = \hat{\mathbf{v}}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

(d)

$$U = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\mathbf{v}_3 = \frac{\mathbf{v}_1 \times \mathbf{v}_2}{\|\mathbf{v}_1 \times \mathbf{v}_2\|} = \frac{\begin{pmatrix} 1 & -1 & 4 \end{pmatrix}^\top \times \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}^\top}{\|\cdot\|} = \frac{\begin{pmatrix} -4 & 4 & 2 \end{pmatrix}^\top}{6} = \begin{pmatrix} -2/3 \\ 2/3 \\ 1/3 \end{pmatrix}$$

$$V = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{pmatrix} = \frac{1}{3\sqrt{2}} \begin{pmatrix} 3 & 1 & -2\sqrt{2} \\ 3 & -1 & 2\sqrt{2} \\ 0 & 4 & \sqrt{2} \end{pmatrix}$$

Answer: $A = U\Sigma V^\top = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \cdot \begin{pmatrix}5 & 0 & 0\\ 0 & 3 & 0\end{pmatrix} \cdot \frac{1}{3\sqrt{2}}\begin{pmatrix}3 & 1 & -2\sqrt{2}\\ 3 & -1 & 2\sqrt{2}\\ 0 & 4 & \sqrt{2}\end{pmatrix}^\top$

Problem 5 (Low-rank approximation; 2 pt)**.** (a) For the matrix A in Problem 4, find a unit vector $\mathbf{x} \in \mathbb{R}^3$ for which $A\mathbf{x}$ has the maximal length α. What is α?

(b) Find the best rank one approximation for the matrix A of Problem 4 in the Frobenius norm.

Solution to the problem 5. .

(a) Let $A = U\Sigma V^\top$ be the SVD of A with $\sigma_1 \geq \cdots \geq \sigma_r$.

$$\mathbf{x}^* : \|A\mathbf{x}^*\| = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\| = \max_{\|\mathbf{x}\|=1} \left\|U^\top A\mathbf{x}\right\| = \max_{\|\mathbf{y}\|=1} \left\|U^\top AV\mathbf{y}\right\| = \max_{\|\mathbf{y}\|=1} \|\Sigma\mathbf{y}\| = \|\Sigma\mathbf{e}_1\| = \sigma_1$$

$$\mathbf{y}^* = \mathbf{e}_1 \qquad \mathbf{x}^* = V\mathbf{y}^* = \mathbf{v}_1$$

So, the maximal length α is the maximum singular value of A: $\sigma_1 = 5$, and the unit vector, for which it is achieved, is the corresponding right singular vector: $\mathbf{v}_1 = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1 & 0\end{pmatrix}^\top$.

Answer: $\mathbf{x}^* = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1 & 0\end{pmatrix}^\top \quad \alpha = 5$

(b) The best rank one approximation in the Frobenius norm is $\sigma_1\mathbf{u}_1\mathbf{v}_1^\top$:

$$\sigma_1\mathbf{u}_1\mathbf{v}_1^\top = \frac{5}{2}\begin{pmatrix}1\\ 1\end{pmatrix}\begin{pmatrix}1 & 1 & 0\end{pmatrix} = \frac{5}{2}\begin{pmatrix}1 & 1 & 0\\ 1 & 1 & 0\end{pmatrix}$$

Problem 6 (Low-rank approximation; 4 pt)**.** Prove that the best rank k approximation A_k of an $m \times n$ matrix A in the Frobenius norm is given by the first k terms in the SVD of A, i.e.,

$$A_k = \sigma_1\mathbf{u}_1\mathbf{v}_1^\top + \cdots + \sigma_k\mathbf{u}_k\mathbf{v}_k^\top.$$

Solution to the problem 6. Let $A = U\Sigma V^\top$. We need to prove that:

$$\|A - A_k\|_F^2 = \min_{\operatorname{rank} B = k} \|A - B\|_F^2$$

Let B be rank-k matrix.

$$\|A - B\|_F^2 = \left\|U\Sigma V^\top - UU^\top BVV^\top\right\|_F^2 = \left\|U(\Sigma - U^\top BV)V^\top\right\|_F^2 = \left\|\Sigma - U^\top BV\right\|_F^2$$

Let $U^\top BV = D + \hat{B}$, where D – diagonal matrix, $\hat{B}$ – off-diagonal.

$$\|A - B\|_F^2 = \sum_i (\Sigma_{ii} - D_{ii})^2 + \sum_{i,j} \hat{B}_{ij}^2$$

To minimize this expression $\hat{B}$ should be zero matrix, so $U^\top BV = D$. Since $\operatorname{rank} B = k \;\Rightarrow\; \operatorname{rank} D \leq k \;\Rightarrow\;$ at most k elements of D are none-zero: $d_i \neq 0,\ i \in K,\ |K| \leq k$.

$$\Rightarrow \quad \|A - B\|_F^2 = \sum_{i \in K} (\sigma_i - d_i)^2 + \sum_{i \notin K} \sigma_i^2$$

From this expression one can see that d_i should be equal to the first k largest singular values to minimize it. So let $\sigma_1 > \cdots > \sigma_r$, then:

$$D = diag\{\sigma_1, \cdots, \sigma_k, 0, \cdots, 0\}$$

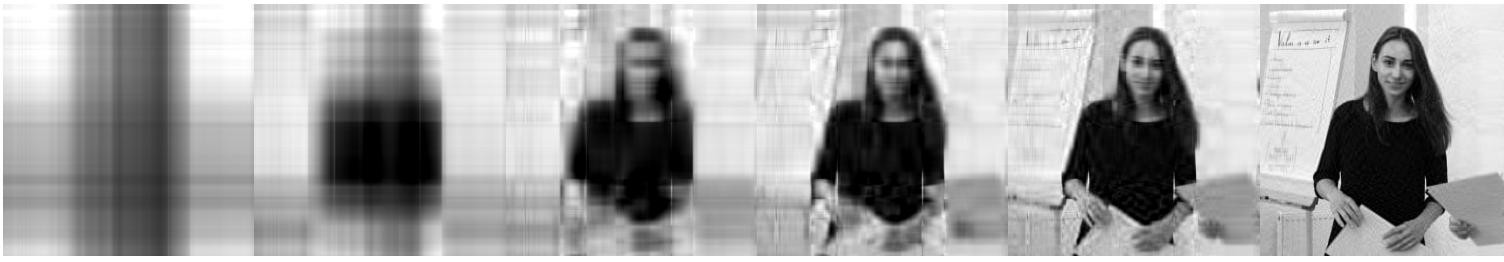
$$\Rightarrow \; B = UDV^\top = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top + \cdots + \sigma_k \mathbf{u}_k \mathbf{v}_k^\top = A_k \qquad \|A - B\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

Problem 7 (SVD and image compression; 5 pt)**.** Take a jpg-picture A of yourselves of reasonable size (say 1000×1000 pixels), perform the SVD (use `Python` or `R` or any other program of your choice) and find the best rank-k approximation of A with $k = 1, 2, 5, 10, 20, 50$. For what k can one recognize the picture? Comment on how much the quality and the size increase along with k.

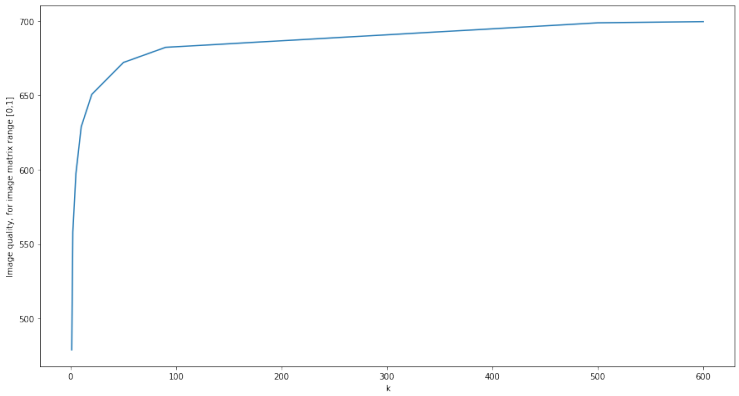
(Hint: one can use the Frobenius distance to the original picture as a quality measure)

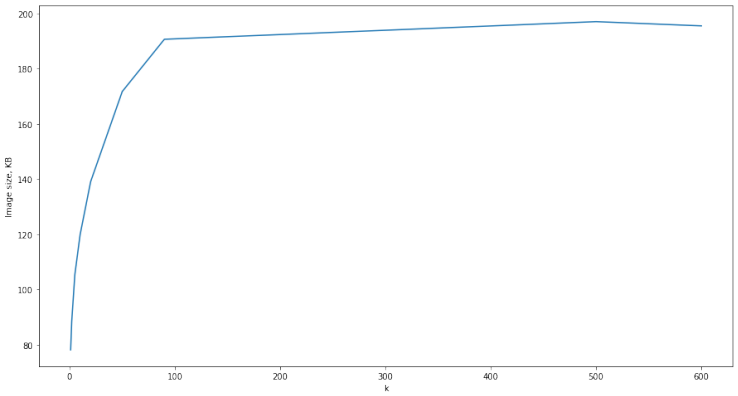
Solution to the problem 7. The original image:

The result of calculating best rank-k approximation of picture for k= $1, 2, 5, 10, 20, 50$:



Below is the dependence between quality (that can be considered e.g. as the Frobenius norm of image minus Frobenius distance between image and its approximation) and k, image size and k:





A person may be recognized starting from k equal to 5-10 already. The quality of image increases considerably when k is small, and then increasing slows down. The size increases even harder as k increases until $k \approx 100$, after that it doesn't change too much.

Problem 8 (Polar decomposition; 3 pt)**.** Find the positive definite square root $S = V\Sigma V^\top$ of $A^\top A$ and its polar decomposition $A = QS$:

$$A = \frac{1}{\sqrt{10}} \begin{pmatrix} 10 & 6 \\ 0 & 8 \end{pmatrix}$$

Solution to the problem 8. .

$$A = \frac{1}{\sqrt{10}} \begin{pmatrix} 10 & 6 \\ 0 & 8 \end{pmatrix} \qquad A^\top A = \begin{pmatrix} 10 & 6 \\ 6 & 10 \end{pmatrix}$$

$$\lambda_1 = 16 \quad \sigma_1 = 4 \qquad \lambda_2 = 20 - 16 = 4 \quad \sigma_2 = 2$$

$$\mathbf{v}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \mathbf{u}_1 = A\mathbf{v}_1/\sigma_1 = \frac{1}{8\sqrt{5}} \begin{pmatrix} 16 \\ 8 \end{pmatrix} = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$\mathbf{v}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad \mathbf{u}_2 = A\mathbf{v}_2/\sigma_2 = \frac{1}{4\sqrt{5}} \begin{pmatrix} 4 \\ -8 \end{pmatrix} = \frac{1}{\sqrt{5}} \begin{pmatrix} 1 \\ -2 \end{pmatrix}$$

$$S = V\Sigma V^\top = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 4 & 4 \\ 2 & -2 \end{pmatrix} = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

$$A = U\Sigma V^\top = UV^\top (V\Sigma V^\top) = QS$$

$$Q = UV^\top = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{10}} \begin{pmatrix} 3 & 1 \\ -1 & 3 \end{pmatrix}$$

Answer: $S = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$; $A = QS = \frac{1}{\sqrt{10}} \begin{pmatrix} 3 & 1 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$

Problem 9 (Pseudoinverses and shortest solutions; 5 pt)**.** (a) Find the SVD and the pseudoinverse $V\Sigma^+ U^\top$ of the matrices

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}.$$

(b) Find the minimum-length solution $\mathbf{x}^+ = A^+\mathbf{b}$ of the equation $A\mathbf{x} = \mathbf{b}$ with

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix}.$$

Solution to the problem 9. Let $\hat{A} = A^T$.

$$\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^\top \;\Rightarrow\; \begin{cases} U = \hat{V} \\ \Sigma = \hat{\Sigma}^\top \text{ for } A \\ V = \hat{U} \end{cases} \Rightarrow\; A^+ = V\Sigma^+U^\top = \hat{U}\Sigma^+\hat{V}^\top$$

(a)

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} \qquad \hat{A} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad \hat{A}^\top\hat{A} = \begin{pmatrix} 4 \end{pmatrix}$$

$$\lambda_1 = 4 \qquad \sigma_1 = 2 \qquad \hat{\Sigma} = \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \end{pmatrix} \;\Rightarrow\; \Sigma^+ = \begin{pmatrix} 1/2 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \hat{\mathbf{v}}_1 = \begin{pmatrix} 1 \end{pmatrix} \qquad \hat{\mathbf{u}}_1 = \frac{\hat{A}\hat{\mathbf{v}}}{\|\cdot\|} = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Let's complete $\hat{U}$ to an ONB of $\mathbb{R}^4$:

$$\hat{U} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \Rightarrow A^+ = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{pmatrix}\begin{pmatrix} 1/2 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \qquad \hat{B} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad \hat{B}^\top\hat{B} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\lambda_1 = \lambda_2 = 1 \qquad \sigma_1 = \sigma_2 = 1 \qquad \hat{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} = \Sigma^+$$

$$\hat{\mathbf{v}}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \hat{\mathbf{v}}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \hat{\mathbf{u}}_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \hat{\mathbf{u}}_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \hat{\mathbf{u}}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$B^+ = \hat{U}\Sigma^+\hat{V}^\top = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \hat{C} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \hat{C}^\top\hat{C} = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\lambda_1 = 2 \qquad \sigma_1 = \sqrt{2} \qquad \hat{\Sigma} = \begin{pmatrix} \sqrt{2} & 0 \\ 0 & 0 \end{pmatrix} \;\Rightarrow\; \Sigma^+ = \begin{pmatrix} 1/\sqrt{2} & 0 \\ 0 & 0 \end{pmatrix}$$

$$\hat{\mathbf{v}}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \hat{\mathbf{v}}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \hat{\mathbf{u}}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \hat{\mathbf{u}}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$C^+ = \hat{U}\Sigma^+\hat{V}^\top = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$$

Answer: $A^+ = \frac{1}{4} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad B^+ = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad C^+ = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$

(b)

$$\mathbf{b} = \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \qquad \hat{A} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \qquad \hat{A}^\top\hat{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

$$\lambda_1 = 4 \quad \lambda_2 = 1 \qquad \sigma_1 = 2 \quad \sigma_2 = 1 \qquad \hat{\Sigma} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \;\Rightarrow\; \Sigma^+ = \begin{pmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} -3 & 1 & 1 \\ 1 & -3 & 1 \\ 1 & 1 & -1 \end{pmatrix} \mathbf{v} = 0 \qquad \hat{\mathbf{v}}_1 = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \qquad \hat{\mathbf{u}}_1 = \frac{\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}}{\|\cdot\|} = \frac{1}{\sqrt{6}} \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 2 \end{pmatrix} \mathbf{v} = 0 \qquad \hat{\mathbf{v}}_2 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} \qquad \hat{\mathbf{u}}_2 = \frac{\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}}{\|\cdot\|} = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix}$$

$$\hat{\mathbf{u}}_3 = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} \text{ (by G-S)} \qquad \hat{V} = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 & \sqrt{2} & \sqrt{3} \\ 1 & \sqrt{2} & -\sqrt{3} \\ 2 & -\sqrt{2} & 0 \end{pmatrix} \qquad \hat{U} = \frac{1}{\sqrt{6}} \begin{pmatrix} 2 & \sqrt{2} & 0 \\ 1 & -\sqrt{2} & \sqrt{3} \\ 1 & -\sqrt{2} & -\sqrt{3} \end{pmatrix}$$

$$A^+ = \frac{1}{6} \begin{pmatrix} 2 & \sqrt{2} & 0 \\ 1 & -\sqrt{2} & \sqrt{3} \\ 1 & -\sqrt{2} & -\sqrt{3} \end{pmatrix} \begin{pmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & \sqrt{2} & \sqrt{3} \\ 1 & \sqrt{2} & -\sqrt{3} \\ 2 & -\sqrt{2} & 0 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 2 & 2 & 0 \\ -1 & -1 & 2 \\ -1 & -1 & 2 \end{pmatrix}$$

$$\mathbf{x}^+ = A^+\mathbf{b} = \frac{1}{4} \begin{pmatrix} 2 & 2 & 0 \\ -1 & -1 & 2 \\ -1 & -1 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \\ 1/2 \end{pmatrix}$$

Answer: $\mathbf{x}^+ = \begin{pmatrix} 1 \\ 1/2 \\ 1/2 \end{pmatrix}$

Problem 10 (Principal component analysis; 3 pt)**.** (a) Convert the matrix of observation A to the zero-mean form and then construct the sample covariance matrix:

$$A = \begin{pmatrix} 19 & 22 & 6 & 3 & 2 & 20 \\ 12 & 6 & 9 & 15 & 13 & 5 \end{pmatrix}$$

(b) Find the principal components of the data in matrix A.

(c) Let x_1, x_2 denote the variables for the two-dimensional data in (a). Find the new variable $y_1 = a_1x_1 + a_2x_2$ such that y_1 has maximum possible variance over the given data. How much of the variance in the data is explained by y_1?

Solution to the problem 10. .

(a) The matrix of observation A is $m \times n$, where $m = 2$ is the dimensionality of the data, $n = 6$ – number of observations (columns). To convert to zero-mean form we calculate mean point across the observations and subtract it from each column of A.

$$A = \begin{pmatrix} 19 & 22 & 6 & 3 & 2 & 20 \\ 12 & 6 & 9 & 15 & 13 & 5 \end{pmatrix} \qquad \mu = \begin{pmatrix} 12 \\ 10 \end{pmatrix} \qquad A_0 = \begin{pmatrix} 7 & 10 & -6 & -9 & -10 & 8 \\ 2 & -4 & -1 & 5 & 3 & -5 \end{pmatrix}$$

$$C = A_0 A_0^\top = \begin{pmatrix} 430 & -135 \\ -135 & 80 \end{pmatrix} \text{ – sample covariance matrix}$$

(b) To find the principal components we find the eigenvalues and eigenvectors of the covariance matrix.

$$\det(C - \lambda) = \lambda^2 - 510\lambda + 16175$$

$$\lambda_1 = 5\left(51 + \sqrt{1954}\right) \approx 476.02 \qquad \lambda_2 = 5\left(51 - \sqrt{1954}\right) \approx 33.98$$

$$\begin{pmatrix} 430 - \lambda & -135 \\ -135 & 80 - \lambda \end{pmatrix} \mathbf{v} = 0$$

Solving the equation for the eigenvalues λ_1, λ_2 we get:

$$\mathbf{v}_1 \approx \begin{pmatrix} 0.95 \\ -0.32 \end{pmatrix} \text{ – the first principal component}$$

$$\mathbf{v}_2 \approx \begin{pmatrix} 0.32 \\ 0.95 \end{pmatrix} \text{ – the second principal component}$$

(c) The new transformed variable with maximum possible variance can be found from the first principal component:

$$y_1 = \mathbf{v}_1^\top \mathbf{x} = 0.95x_1 - 0.32x_2 \qquad \lambda_1 = 476.02 \text{ – corresponds to the variance of } y_1$$

And the fraction of the total variance is:

$$\frac{\lambda_1}{\sum_{i=1}^{2} \lambda_i} = 93.3\%$$

So, the new transformed and restricted data would be:

$$Y_1 = \begin{pmatrix} 14.11 & 18.89 & 2.78 & -2 & -2.3 & 17.32 \end{pmatrix}$$

And, indeed:

$$\frac{Var(Y_1)}{\sum_{i=1}^{2} Var(A_i)} = \frac{79.34}{71.67 + 13.33} = 93.3\%$$

, where A_i corresponds to observations of the i^{th} component (row) of the data.

Problem 11 (PCA; 5 pt)**.** (a) Simulate $N = 100$ data (x_k, y_k) from the two-dimensional Gaussian (normal) distribution $\mathcal{N}(\mu_1, \mu_2; \sigma_1^2, \sigma_2^2, \rho)$ with $\mu_1 = 1$, $\mu_2 = 2$, $\sigma_1 = 4$, $\sigma_2 = 9$, $\rho = \frac{1}{3}$.

Hint: Can you do this easily if $\rho = 0$? If $(Z_1, Z_2)^\top \sim \mathcal{N}(0, 0; 1, 1, 0)$, show that $(X, Y)^\top$ with $X = \mu_1 + \sigma_1 Z_1$ and $Y = \mu_2 + \sigma_2 \rho Z_1 + \sqrt{1 - \rho^2}\sigma_2 Z_2$ has the required distribution

(b) Form the empirical covariance matrix C for the data simulated and find its eigenvalues and eigenvectors.

(c) Perform the PCA on the data generated. Calculate the variance along the first component; what fraction of the total variance does it include?

(d) Predict how the above fraction depends on ρ and confirm your reasoning numerically.

Solution to the problem 11. . Since Z_1, Z_2 are normally distributed, and X, Y are given from the linear transformations of Z_1, Z_2, it's also normally distributed. Let's find the parameters:

$$EX = E(\mu_1 + \sigma_1 Z_1) = \mu_1 \quad EY = E(\mu_2 + \sigma_2 \rho Z_1 + \sigma_2 \sqrt{1 - \rho^2} Z_2) = \mu_2$$

$$Var(X) = Var(\mu_1 + \sigma_1 Z_1) = \sigma_1^2 Var(Z_1) = \sigma_1^2$$

$$Var(Y) = Var\left(\mu_2 + \sigma_2(\rho Z_1 + \sqrt{1 - \rho^2} Z_2)\right) = \sigma_2^2 Var\left(\rho Z_1 + \sqrt{1 - \rho^2} Z_2\right) =$$

$$= \| \ Z_1 \text{ and } Z_2 \text{ are independent } \| = \sigma_2^2(\rho^2 Var(Z_1) + (1 - \rho^2) Var(Z_2))$$

$$= \sigma_2^2(\rho^2 + 1 - \rho^2) = \sigma_2^2$$

$$Corr(X, Y) = \frac{Cov(X, Y)}{\sigma_1 \sigma_2} = \frac{1}{\sigma_1 \sigma_2} Cov\left(\mu_1 + \sigma_1 Z_1, \ \mu_2 + \sigma_2(\rho Z_1 + \sqrt{1 - \rho^2} Z_2)\right) =$$

$$= \frac{\sigma_1 \sigma_2}{\sigma_1 \sigma_2} Cov\left(Z_1, \ \rho Z_1 + \sqrt{1 - \rho^2} Z_2\right) = (\rho Var(Z_1) + \sqrt{1 - \rho^2} Cov(Z_1, Z_2)) =$$

$$= | \ Z_1 \text{ and } Z_2 \text{ are independent } | = \rho$$

So (X, Y) has the required distribution. See the code with its output below:

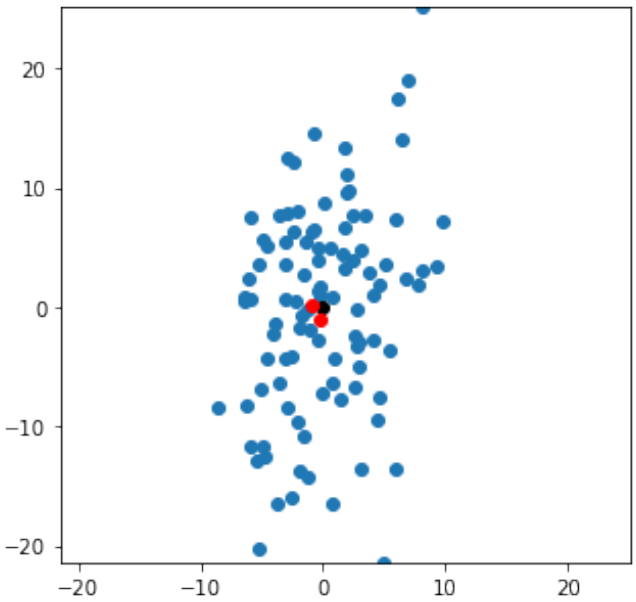
```
In [1]: import numpy as np
        import scipy.stats as stats
        from scipy import linalg as LA
        from matplotlib import pyplot as plt
        import matplotlib as mpl

        def sample_mvn2D_from_1D(mu1, mu2, sigma1, sigma2, rho, N):
```

```
        x0 = np.random.normal(size=N)
        y0 = np.random.normal(size=N)
        x = mu1 + sigma1*x0
        y = mu2 + sigma2*(rho*x0 + np.sqrt(1-rho**2)*y0)
        A = np.hstack((x.reshape(-1,1), y.reshape(-1,1)))
        return A

    def visualize2D(A, cmap='blue'):
        A = A.reshape((-1,2))
        plt.scatter(A[:,0], A[:,1])
        lmax, lmin = np.max(A), np.min(A)
        plt.xlim(lmin,lmax)
        plt.ylim(lmin,lmax)

    def visualizeEVec(V):
        V = np.array(V)
        plt.scatter(0,0, c='k')
        plt.scatter(V[0,:], V[1,:], c='red')

    def PCA(A, out=False):
        mean_point = np.mean(A, 0, keepdims=True)
        A0 = A - mean_point
        C = A0.T.dot(A0)
        if out:
            print('Covariance matrix:\n{}'.format(C))
        l, v = LA.eig(C)
        l = l.real
        ids = np.argsort(-l)
        lambdas, V = l[ids], v[:,ids]
        principal_components = list(V.T)
        if out:
            print('Eigenvalues & eigenvectors')
            for i, (l, v) in enumerate(list(zip(lambdas, principal_components))):
                print('l_{} = {:.2f}\tv_{} = {}'.format(i, l, i, v.flatten()))
        return principal_components

    def projectPoints(A, v):
        mean_point = np.mean(A, 0, keepdims=True)
        A0 = A - mean_point
        P = v.dot(v.T) / (v.T.dot(v))
        PA = (P.dot(A0.T)).T + mean_point
        return PA
```

In [2]:
```
N = 100
mu1, mu2 = 1, 2
sigma1, sigma2 = 4, 9
```

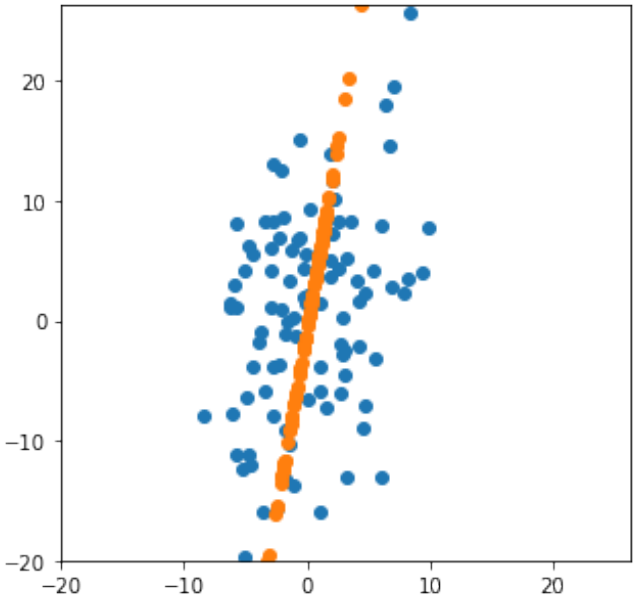
```
        rho = 1/3

        mpl.rcParams['figure.figsize'] = (5,5)
        A = sample_mvn2D_from_1D(mu1, mu2, sigma1, sigma2, rho, N)
        mean_point = np.mean(A, 0, keepdims=True)
        A0 = A - mean_point
        pc = PCA(A, out=True)

        print('\nCentered data points (blue) with calculated principal components (red):')
        visualize2D(A0)
        visualizeEVec(pc)
        plt.show()

        v1 = pc[0].reshape(-1,1)
        newA = projectPoints(A, v1)
        total_variance = np.sum(np.var(A, 0))
        variance1 = np.mean(np.sum((newA-mean_point)**2, axis=1))
        fraction = variance1 / total_variance * 100
        print('\n{} principal component:'.format(1))
        print('\tPoints projected on the component (orange):')
        visualize2D(A)
        visualize2D(newA)
        plt.show()
        print('\tVariance along the component: {:.2f}'.format(variance1))
        print('\tThe fration of the total variance: {:.2f} / {:.2f} = {:.2f}%'.format(variar
```

```
Covariance matrix:
[[1675.75323616  997.42117194]
 [ 997.42117194 7602.66696436]]
Eigenvalues & eigenvectors
l_0 = 7766.02        v_0 = [-0.16161994 -0.98685308]
l_1 = 1512.40        v_1 = [-0.98685308  0.16161994]

Centered data points (blue) with calculated principal components (red):
```



1 principal component:

Points projected on the component (orange):

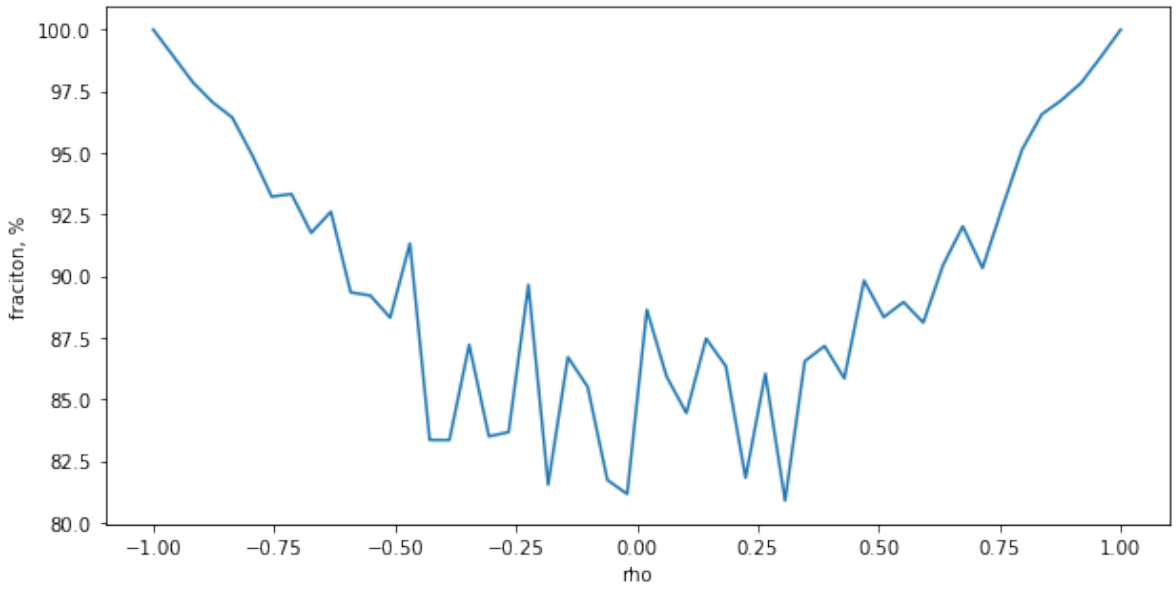


Variance along the component: 77.66

The fration of the total variance: 77.66 / 92.78 = 83.70%

```
In [3]: rhos = np.linspace(-1,1,50)
        fractions = []
        for rho in rhos:
            A = sample_mvn2D_from_1D(mu1, mu2, sigma1, sigma2, rho, N)
            mean_point = np.mean(A, 0, keepdims=True)
            total_variance = np.sum(np.var(A, 0))
            pc = PCA(A)
            v1 = pc[0].reshape(-1,1)
            newA = projectPoints(A, v1)
            variance1 = np.mean(np.sum((newA-mean_point)**2, axis=1))
            fraction = variance1 / total_variance * 100
            fractions.append(fraction)
```

```
In [4]: mpl.rcParams["figure.figsize"] = (10,5)
        plt.plot(rhos, fractions)
        plt.xlabel('rho')
        plt.ylabel('fraciton, %')
        plt.show()
```



So, the variance fraction increases with increasing of absolute value of ρ.

Problem 12 (PCA in many dimensions; 8 pt)**.** This is a 10-dimensional analogue of Problem 11.

(a) Simulate $N = 100$ data $\mathbf{x}_j = (x_1^{(j)}, x_2^{(j)}, \ldots, x_{10}^{(j)})^\top$ from the Gaussian (normal) distribution $\mathcal{N}(\mathbf{0}, \Sigma)$ with $\Sigma = I + \mathbf{u}\mathbf{u}^\top$, where $\mathbf{u} = (1; -2; 3; \ldots; -10)^\top$

Hint: if you factorize $\Sigma = LL^\top$ with a lower-triangular L (Cholesky factorization), simulate the standard Gaussian vectors $\mathbf{z}_j = (z_1^{(j)}, z_2^{(j)}, \ldots, z_{10}^{(j)})^\top$ (ie, with independent components of variance 1), then $\mathbf{x} = L\mathbf{z} + \boldsymbol{\mu}$ will follow $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Justify before using that!

(b) Form the empirical covariance matrix C for the data simulated and find its eigenvalues and eigenvectors.

(c) Perform the PCA on the data generated. Calculate the variance along the several first component; what fraction of the total variance does it include?

(d) Predict how the above fraction depends on the shape of the set $\{\mathbf{x} \mid \mathbf{x}^\top \Sigma \mathbf{x} = 1\}$ and confirm your reasoning numerically.

Hint: certainly you will need to use R, Python or MatLab libraries to solve this problem!

Solution to the problem 12. . Let Z be a $d \times N$ data matrix ($d = 10, N = 100$) generated from the standard Gaussian distribution $\mathcal{N}(\mathbf{0}_d, I_d)$. Let Σ be a covariance matrix and L – a lower-triangular matrix from the Cholesky decomposition $\Sigma = LL^\top$, $\mu = \begin{pmatrix}\mu_1 & \cdots & \mu_d\end{pmatrix}^\top$. Since the linear transformation of the normal random variable is also a normal random variable and:

$$X = LZ + \mu \;\Rightarrow\; EX = LE(Z) + \mu = L\underline{0}_d + \mu = \mu$$

$$Var(X) = E((X - EX)(X - EX)^\top) = E(LZ(LZ)^\top) = LE(ZZ^\top)L^\top = LI_dL^\top = \Sigma$$

we can simulate X sampling using Z, L and μ: $X = LZ + \mu$.

$$\{\mathbf{x} \mid \mathbf{x}^\top \Sigma \mathbf{x} = 1\} = \{\mathbf{x} \mid (L^\top \mathbf{x})^\top L^\top \mathbf{x} = 1\} = \{\mathbf{x} = L^{-\top}\mathbf{y} \mid \|\mathbf{y}\| = 1\}$$

Intuitively it seems that for the first principal components the fraction of the total variance will be more with such covariance matrix that skews the unit ball. We can look at the projections of some points of the set in 2D using $\mathbf{y}$ (such that $\mathbf{y}$ has only 2 non-zero coordinates). See the code with the output below:

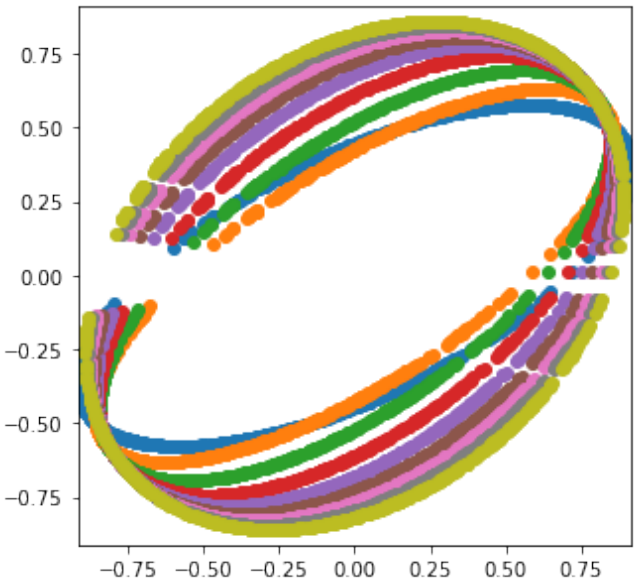
```
In [1]: import numpy as np
        import scipy.stats as stats
        from scipy import linalg as LA
        from matplotlib import pyplot as plt
        import matplotlib as mpl


        def sample_mvn_from1D(Mu, Sigma, N):
            d = Sigma.shape[0]
            Z = np.random.normal(size=(N,d))
            L = LA.cholesky(Sigma, lower=True)
            X = L.dot(Z.T).T + Mu
            return X

        def variance_along_axis(A, v):
            mean_point = np.mean(A, 0, keepdims=True)
            A_norm = A - mean_point
            P = v.dot(v.T) / (v.T.dot(v))
            PA = (P.dot(A_norm.T)).T
            var = np.mean(np.sum(PA**2, axis=1))
            return var
```

```
        def PCA(A, n_components, out=False):
            mean_point = np.mean(A, 0, keepdims=True)
            A0 = A - mean_point
            C = A0.T.dot(A0)
            if out:
                print('Covariance matrix:\n{}'.format(C))
            l, v = LA.eig(C)
            l = l.real
            ids = np.argsort(-l)
            lambdas, principal_components = l[ids], v[:,ids]
            if out:
                print('\nEigenvalues & eigenvectors:'.format(n_components))
                for i, (l, v) in enumerate(list(zip(lambdas, principal_components))):
                    print('l_{} = {:.2f}\tv_{} = {}'.format(i, l, i, v.flatten()))
            return principal_components[:,:n_components]

        def visualize_set(Sigma):
            print('Visualization of the projections of some points \n\
        of the set {x | (L^T x)^T L^T x = 1}.')
            print('Different colors correspond to different (2x2) parts of L:')
            LT = LA.cholesky(Sigma, lower=True).T
            LTinv = LA.inv(LT)
            N = 1000
            xs = (np.random.rand(N//2,1) - .5) * 2
            xs = np.vstack((xs, (1-np.random.rand(N//2,1) - .5) * 2))
            ys = np.sign((np.random.rand(N,1) - .5) * 2) * np.sqrt(1 - xs**2)
            Y = np.hstack((xs,ys))
            lmax, lmin = 0, 0
            for i in range(LT.shape[0]-1):
                X = (LTinv[i:i+2,i:i+2].dot(Y.T)).T
                visualize2D(X)
                lmax, lmin = max(np.max(X), lmax), min(np.min(X), lmin)
                plt.xlim(lmin,lmax)
                plt.ylim(lmin,lmax)

        def visualize2D(A, cmap='blue'):
            A = A.reshape((-1,2))
            plt.scatter(A[:,0], A[:,1])
            lmax, lmin = np.max(A), np.min(A)
            plt.xlim(lmin,lmax)
            plt.ylim(lmin,lmax)
```

In [2]:
```
        mpl.rcParams['figure.figsize'] = (5,5)
        N = 100
        d = 10
```

```
        Mu = np.zeros(d)
        u = np.array([(-1)**i*(i+1) for i in range(10)]).reshape(-1,1)
        Sigma = np.eye(d)+u.dot(u.T)
        visualize_set(Sigma)
        plt.show()
        A = sample_mvn_from1D(Mu, Sigma, N)
        n_components = 5
        total_variance = np.sum(np.var(A, 0))
        pc = PCA(A, n_components, out=True)
        print()
        for i in range(n_components):
            v = pc[:,i].reshape(-1,1)
            variance = variance_along_axis(A, v)
            fraction = variance / total_variance * 100
            print('{} principal component:'.format(i+1, v.flatten()))
            print('\tVariance along the component: {:.2f}'.format(variance))
            print('\tThe fraction of the total variance: {:.2f} / {:.2f} = {:.2f}%'.format(v
```

```
Visualization of the projections of some points
of the set {x | (L^T x)^T L^T x = 1}.
Different colors correspond to different (2x2) parts of L:
```



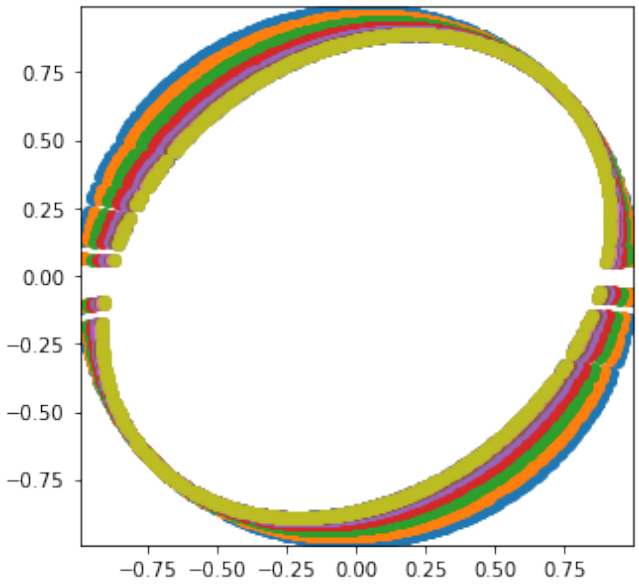

```
Covariance matrix:
[[  214.78971657  -201.24955179   300.72160209  -385.72372796
    529.6520468   -611.37256151   705.68227067  -785.75569745
    868.94607174  -991.48615739]
```

```
 [ -201.24955179   537.03948453  -657.00300317   820.12962187
  -1089.48325768  1274.24170173 -1454.40059324  1647.7134575
  -1849.45677325  2036.21055661]
 [  300.72160209  -657.00300317  1076.20879342 -1180.01687008
   1598.14637295 -1870.08046825  2176.96746085 -2470.6720384
   2749.01159692 -3057.06573498]
 [ -385.72372796   820.12962187 -1180.01687008  1607.81048663
  -2000.03664061  2361.37559348 -2731.42892245  3069.48603077
  -3451.32196591  3853.66346444]
 [  529.6520468  -1089.48325768  1598.14637295 -2000.03664061
   2767.26495399 -3141.47549752  3655.56988208 -4090.77292491
   4620.36946533 -5128.81104216]
 [ -611.37256151  1274.24170173 -1870.08046825  2361.37559348
  -3141.47549752  3791.3181555  -4296.12885672  4837.60632513
  -5445.56903835  6042.94906325]
 [  705.68227067 -1454.40059324  2176.96746085 -2731.42892245
   3655.56988208 -4296.12885672  5057.15912953 -5602.80979281
   6303.60656621 -6994.7646079 ]
 [ -785.75569745  1647.7134575  -2470.6720384   3069.48603077
  -4090.77292491  4837.60632513 -5602.80979281  6400.591782
  -7063.53973294  7869.54292829]
 [  868.94607174 -1849.45677325  2749.01159692 -3451.32196591
   4620.36946533 -5445.56903835  6303.60656621 -7063.53973294
   8075.30042715 -8865.16181702]
 [ -991.48615739  2036.21055661 -3057.06573498  3853.66346444
  -5128.81104216  6042.94906325 -6994.7646079   7869.54292829
  -8865.16181702  9968.56548489]]

Eigenvalues & eigenvectors:
l_0 = 38553.62        v_0 = [ 0.05069188  0.10630869  0.25196444 -0.71117634  0.36112074
                              0.0940609  0.26835339 -0.32456015 -0.25349071 -0.18974959]
l_1 = 158.97        v_1 = [-0.10576754  0.39091391 -0.58590637 -0.14809867  0.16127227
                             0.17173819 -0.23807858 -0.21036262 -0.12280814  0.5470481 ]
l_2 = 134.95        v_2 = [ 0.15746259 -0.65532099 -0.25118828 -0.00585876  0.15823129
                            -0.44684209  0.07969758 -0.44932539  0.09009407  0.20387032]
l_3 = 134.29        v_3 = [-0.19755108 -0.14206053 -0.41570398  0.09890565  0.60579719
                             0.16523503  0.33971232  0.39989432  0.16673445 -0.24655041]
l_4 = 121.58        v_4 = [ 0.26361037 -0.05062176  0.38540734 -0.06938088  0.46176641
                            -0.21528201 -0.28926165  0.47068041 -0.06146408  0.453905  ]
l_5 = 107.77        v_5 = [-0.31053846 -0.01581623 -0.154455   -0.0314059   0.1211
                            -0.39028824 -0.57065775  0.03795466 -0.41308429 -0.4623348 ]
l_6 = 89.88        v_6 = [ 0.3596452  -0.018248   -0.00080067 -0.02997386  0.21191232
                             0.33795537 -0.53581496 -0.23252026  0.52900921 -0.29881183]
l_7 = 77.20        v_7 = [-0.40447002  0.42244607  0.28522643  0.24999423  0.28817262
                            -0.38844971  0.09066107 -0.31512673  0.41344372  0.04117586]
l_8 = 63.05        v_8 = [ 0.45525776  0.20228606  0.02466264  0.59015555  0.27238072
```

```
                          0.07480811  0.1359388  -0.26071837 -0.47121571 -0.12101123]
l_9 = 54.75         v_9 = [-0.50625863 -0.40463804  0.32088023  0.21213012  0.14669942
                           0.51508598 -0.16257261 -0.21041461 -0.19363921  0.19228197]

1 principal component:
        Variance along the component: 385.54
        The fraction of the total variance: 385.54 / 394.96 = 97.61%
2 principal component:
        Variance along the component: 1.59
        The fraction of the total variance: 1.59 / 394.96 = 0.40%
3 principal component:
        Variance along the component: 1.35
        The fraction of the total variance: 1.35 / 394.96 = 0.34%
4 principal component:
        Variance along the component: 1.34
        The fraction of the total variance: 1.34 / 394.96 = 0.34%
5 principal component:
        Variance along the component: 1.22
        The fraction of the total variance: 1.22 / 394.96 = 0.31%
```

```
In [3]: mpl.rcParams['figure.figsize'] = (5,5)
        N = 100
        d = 10
        Mu = np.zeros(d)
        u = np.array([(-1)**i*(i+1) for i in range(10)]).reshape(-1,1)
        Sigma = np.eye(d)+u.dot(u.T) / 100
        visualize_set(Sigma)
        plt.show()
        A = sample_mvn_from1D(Mu, Sigma, N)
        n_components = 5
        total_variance = np.sum(np.var(A, 0))
        pc = PCA(A, n_components, out=False)
        print()
        for i in range(n_components):
            v = pc[:,i].reshape(-1,1)
            variance = variance_along_axis(A, v)
            fraction = variance / total_variance * 100
            print('{} principal component:'.format(i+1, v.flatten()))
            print('\tVariance along the component: {:.2f}'.format(variance))
            print('\tThe fraction of the total variance: {:.2f} / {:.2f} = {:.2f}%'.format(v
```

```
Visualization of the projections of some points
of the set {x | (L^T x)^T L^T x = 1}.
Different colors correspond to different (2x2) parts of L:
```

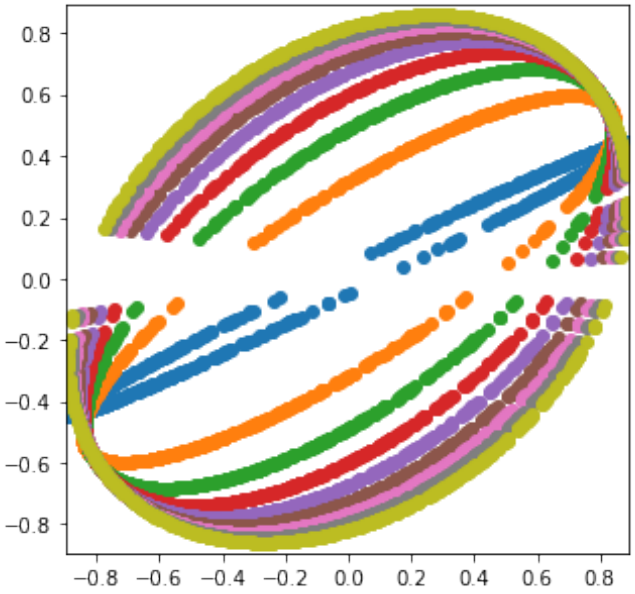



```
1 principal component:
        Variance along the component: 4.32
        The fraction of the total variance: 4.32 / 12.61 = 34.24%
2 principal component:
        Variance along the component: 1.37
        The fraction of the total variance: 1.37 / 12.61 = 10.84%
3 principal component:
        Variance along the component: 1.17
        The fraction of the total variance: 1.17 / 12.61 = 9.29%
4 principal component:
        Variance along the component: 1.13
        The fraction of the total variance: 1.13 / 12.61 = 8.92%
5 principal component:
        Variance along the component: 0.97
        The fraction of the total variance: 0.97 / 12.61 = 7.66%
```

```
In [4]: mpl.rcParams['figure.figsize'] = (5,5)
        N = 100
        d = 10
        Mu = np.zeros(d)
        u = np.array([(-1)**i*(i+1) for i in range(10)]).reshape(-1,1)
        Sigma = np.eye(d)+u.dot(u.T) * 100
        visualize_set(Sigma)
        plt.show()
        A = sample_mvn_from1D(Mu, Sigma, N)
```

```
        n_components = 5
        total_variance = np.sum(np.var(A, 0))
        pc = PCA(A, n_components, out=False)
        print()
        for i in range(n_components):
            v = pc[:,i].reshape(-1,1)
            variance = variance_along_axis(A, v)
            fraction = variance / total_variance * 100
            print('{} principal component:'.format(i+1, v.flatten()))
            print('\tVariance along the component: {:.2f}'.format(variance))
            print('\tThe fraction of the total variance: {:.2f} / {:.2f} = {:.2f}%'.format(v
```

```
Visualization of the projections of some points
of the set {x | (L^T x)^T L^T x = 1}.
Different colors correspond to different (2x2) parts of L:
```




```
1 principal component:
        Variance along the component: 38779.78
        The fraction of the total variance: 38779.78 / 38788.46 = 99.98%
2 principal component:
        Variance along the component: 1.44
        The fraction of the total variance: 1.44 / 38788.46 = 0.00%
3 principal component:
        Variance along the component: 1.32
        The fraction of the total variance: 1.32 / 38788.46 = 0.00%
```

```
4 principal component:
        Variance along the component: 1.22
        The fraction of the total variance: 1.22 / 38788.46 = 0.00%
5 principal component:
        Variance along the component: 1.00
        The fraction of the total variance: 1.00 / 38788.46 = 0.00%
```

So, in the first example (corresponding to the matrix from the task) we can see that in some dimensions the set of points looks like a skewed ellipse, and the fraction for the 1 component is 97.61%. In the second example the covariance matrix is close to I_d, so the set is close to a ball, and the fraction is not so big – 34.24%. In the last example there are two dimensions in which the ellipse is highly skewed, and the fraction is also very high – 99.98%.

Problem 13 (Jacobi and Gauss–Seidel iteration scheme; 6 pt)**.** Use the Jacobi and Gauss–Seidel methods to solve the 3×3 system $A\mathbf{x} = \mathbf{b}$ with

$$A = \alpha I_3 + \mathbf{u}\mathbf{u}^\top, \qquad \mathbf{b} = \mathbf{u},$$

where $\mathbf{u} = (1, -1, 1)^\top$.

(a) For what α do the methods work?

(b) Write the corresponding iteration scheme for the Jacobi method and find the solution starting with $\mathbf{x}^{(1)} = \mathbf{0}$. How many iteration are required to achieve 0.001 accuracy?

(c) Write the corresponding iteration scheme for the Gauss–Seidel method and find the solution starting with $\mathbf{x}^{(1)} = \mathbf{0}$. How many iteration are required to achieve 0.001 accuracy?

Solution to the problem 13.

$$A = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 1 & -1 & 0 \end{pmatrix} + \begin{pmatrix} 1+\alpha & 0 & 0 \\ 0 & 1+\alpha & 0 \\ 0 & 0 & 1+\alpha \end{pmatrix} + \begin{pmatrix} 0 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} = L + D + U$$

For the Jacobi method $\mathbf{x}^{(k+1)} = B_J \mathbf{x}^{(k)} + \mathbf{b}_J$

$$B_J = -\frac{1}{1+\alpha} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix} = \frac{1}{1+\alpha} \begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}$$

$$\mathbf{b}_J = \frac{1}{1+\alpha} \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$$

So

$$\mathbf{x}^{(k+1)} = \frac{1}{1+\alpha} \left(\begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix} \mathbf{x}^{(k)} + \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \right)$$

To check what α should be to converge, we need to find eigenvalues of B_J:

$$-\lambda^3 + \frac{3}{(\alpha+1)^2}\lambda - \frac{2}{(\alpha+1)^3} = 0 \;\Leftrightarrow\; \left(\lambda - \frac{1}{\alpha+1}\right)^2\left(\lambda + \frac{2}{\alpha+1}\right) = 0$$

$$\lambda_{1,2} = \frac{1}{\alpha+1} \qquad \lambda_3 = -\frac{2}{\alpha+1}$$

The maximum absolute value of eigenvalue must be less than $1 \Rightarrow -0.5 < \frac{1}{\alpha+1} < 0.5 \;\Rightarrow\; \alpha+1 \in (-\infty;-2)\cup(2;\infty) \;\Rightarrow\; \alpha \in (-\infty;-3)\cup(1;\infty)$ – this are the possible values of α, for which the Jacobi method works.
For the Gauss-Seidel method $\mathbf{x}^{(k+1)} = B_{GS}\mathbf{x}^{(k)} + \mathbf{b}_{GS}$

$$B_{GS} = -\begin{pmatrix} 1+\alpha & 0 & 0 \\ -1 & 1+\alpha & 0 \\ 1 & -1 & 1+\alpha \end{pmatrix}^{-1}\begin{pmatrix} 0 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} =$$

$$= \frac{1}{(1+\alpha)^3}\begin{pmatrix} (1+\alpha)^2 & 0 & 0 \\ 1+\alpha & (1+\alpha)^2 & 0 \\ -\alpha & 1+\alpha & (1+\alpha)^2 \end{pmatrix}\begin{pmatrix} 0 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} = \frac{1}{(1+\alpha)^3}\begin{pmatrix} 0 & (1+\alpha)^2 & -(1+\alpha)^2 \\ 0 & 1+\alpha & \alpha(1+\alpha) \\ 0 & -\alpha & 1+2\alpha \end{pmatrix}$$

$$b_{GS} = \frac{1}{(1+\alpha)^3}\begin{pmatrix} (1+\alpha)^2 & 0 & 0 \\ 1+\alpha & (1+\alpha)^2 & 0 \\ -\alpha & 1+\alpha & (1+\alpha)^2 \end{pmatrix}\begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} = \frac{1}{(1+\alpha)^3}\begin{pmatrix} (1+\alpha)^2 \\ -\alpha(1+\alpha) \\ \alpha^2 \end{pmatrix}$$

So

$$\mathbf{x}^{(k+1)} = \frac{1}{(1+\alpha)^3}\left(\begin{pmatrix} 0 & (1+\alpha)^2 & -(1+\alpha)^2 \\ 0 & 1+\alpha & \alpha(1+\alpha) \\ 0 & -\alpha & 1+2\alpha \end{pmatrix}\mathbf{x}^{(k)} + \begin{pmatrix} (1+\alpha)^2 \\ -\alpha(1+\alpha) \\ \alpha^2 \end{pmatrix}\right)$$

To study the methods convergence we can also check when A is row diagonally dominant:
$|1+\alpha| > 2 \;\Rightarrow\; \alpha \in (-\infty;-3)\cup(1;\infty)$. For these α the methods work.
The provided code with its output is below:

```
In [1]: import numpy as np
        import scipy as sp
        import scipy.stats as stats
        from scipy import linalg as LA
        from math import sqrt

        class IterativeMethod():
            def __init__(self, A, b):
                self.A = A
                self.b = b
                self.m, self.n = A.shape

                self.LU = self.A * (1 - np.eye(min(*A.shape)))
                self.U = self.LU.copy()
                for i in range(self.m):
                    for j in range(i):
```

```
                self.U[i,j] = 0
        self.L = self.LU - self.U
        self.D = self.A - self.LU
        self.Dinv = LA.inv(self.D)

        self.BJ = - self.Dinv.dot(self.LU)
        self.bJ = self.Dinv.dot(self.b)

    def stop_criteria(self, x_prev, x_next, eps=1e-5):
        difference = LA.norm(x_prev - x_next) / (LA.norm(x_prev)+eps)
        accuracy = self.accuracy(x_next)
        if self.out:
            print('\trel. diff. = {:.4f}, acc. = {:.4f}'\
                  .format(difference, accuracy))
#         return difference < self.diff_eps
        return accuracy < self.acc_eps

    def accuracy(self, x):
        return LA.norm(self.A.dot(x) - self.b)

    def check_B(self):
        eigvalues, eigvectors = LA.eig(self.BJ)
        print(max(np.abs(eigvalues)))

    def run(self, x0, out=True, diff_eps=1e-5, acc_eps=1e-3, maxiter=1e3):
        self.out = out
        self.diff_eps = diff_eps
        self.acc_eps = acc_eps
        self.maxiter = maxiter
        if self.out:
            print('\tx_{} = {}'.format(0, x0.flatten()))
        x_prev = x0
        x_next = self.iteration(x_prev)
        it = 1
        while not self.stop_criteria(x_prev, x_next) and it < self.maxiter:
            it += 1
            x_prev = x_next
            x_next = self.iteration(x_prev)
            if out:
                print('\tx_{} = {}'.format(it-1, x_prev.flatten()))
        self.finalize(x_next, it)
        return self.x

    def finalize(self, x, it):
        self.it = it
        if self.it < self.maxiter:
```

```
                self.x = x
                self.acc = self.accuracy(x)
                print('The method converged in {} it. to x* = {}'\
                      .format(self.it, self.x.flatten()))
                print('The accuracy is {:.4f}'.format(self.acc))
            else:
                self.x = np.nan
                print('The method didn\'t converge.')

    class Jacobi(IterativeMethod):

        def iteration(self, x_prev):
            x_next = self.BJ.dot(x_prev) + self.bJ
            return x_next


    class GaussSeidel(IterativeMethod):

        def iteration(self, x_prev):
            x_next = np.zeros_like(x_prev)
            Dinv = LA.inv(self.D)
            for i in range(x_next.shape[0]):
                x_next[i] = Dinv[i,i] * (self.b[i] - self.L[i,:].dot(x_next)
                                                   - self.U[i,:].dot(x_prev))
            return x_next
```

In [2]:
```
u = np.array([1,-1,1]).reshape((-1,1))
alpha = 2
A = alpha * np.eye(3) + u.dot(u.T)
b = u
x0 = np.zeros((3,1))
```

In [3]:
```
J = Jacobi(A, b)
x = J.run(x0, out=True, maxiter=1000, acc_eps=0.001)
```

```
x_0 = [0. 0. 0.]
rel. diff. = 57735.0269, acc. = 1.1547
x_1 = [ 0.33333333 -0.33333333  0.33333333]
rel. diff. = 0.6667, acc. = 0.7698
x_2 = [ 0.11111111 -0.11111111  0.11111111]
rel. diff. = 1.3333, acc. = 0.5132
x_3 = [ 0.25925926 -0.25925926  0.25925926]
rel. diff. = 0.3809, acc. = 0.3421
x_4 = [ 0.16049383 -0.16049383  0.16049383]
rel. diff. = 0.4102, acc. = 0.2281
x_5 = [ 0.22633745 -0.22633745  0.22633745]
rel. diff. = 0.1939, acc. = 0.1521
```

```
        x_6 = [ 0.1824417 -0.1824417  0.1824417]
        rel. diff. = 0.1604, acc. = 0.1014
        x_7 = [ 0.21170553 -0.21170553  0.21170553]
        rel. diff. = 0.0922, acc. = 0.0676
        x_8 = [ 0.19219631 -0.19219631  0.19219631]
        rel. diff. = 0.0677, acc. = 0.0451
        x_9 = [ 0.20520246 -0.20520246  0.20520246]
        rel. diff. = 0.0423, acc. = 0.0300
        x_10 = [ 0.19653169 -0.19653169  0.19653169]
        rel. diff. = 0.0294, acc. = 0.0200
        x_11 = [ 0.2023122 -0.2023122  0.2023122]
        rel. diff. = 0.0190, acc. = 0.0133
        x_12 = [ 0.19845853 -0.19845853  0.19845853]
        rel. diff. = 0.0129, acc. = 0.0089
        x_13 = [ 0.20102765 -0.20102765  0.20102765]
        rel. diff. = 0.0085, acc. = 0.0059
        x_14 = [ 0.1993149 -0.1993149  0.1993149]
        rel. diff. = 0.0057, acc. = 0.0040
        x_15 = [ 0.20045673 -0.20045673  0.20045673]
        rel. diff. = 0.0038, acc. = 0.0026
        x_16 = [ 0.19969551 -0.19969551  0.19969551]
        rel. diff. = 0.0025, acc. = 0.0018
        x_17 = [ 0.20020299 -0.20020299  0.20020299]
        rel. diff. = 0.0017, acc. = 0.0012
        x_18 = [ 0.19986467 -0.19986467  0.19986467]
        rel. diff. = 0.0011, acc. = 0.0008
The method converged in 19 it. to x* = [ 0.20009022 -0.20009022  0.20009022]
The accuracy is 0.0008
```

```
In [4]: GS = GaussSeidel(A, b)
        x = GS.run(x0, out=True, maxiter=1000, acc_eps=0.001)

        x_0 = [0. 0. 0.]
        rel. diff. = 42713.1948, acc. = 0.3989
        x_1 = [ 0.33333333 -0.22222222  0.14814815]
        rel. diff. = 0.3074, acc. = 0.0566
        x_2 = [ 0.20987654 -0.21399177  0.1920439 ]
        rel. diff. = 0.0496, acc. = 0.0082
        x_3 = [ 0.19798811 -0.20332266  0.19956307]
        rel. diff. = 0.0089, acc. = 0.0023
        x_4 = [ 0.19903809 -0.20046628  0.20016521]
        rel. diff. = 0.0025, acc. = 0.0006
The method converged in 5 it. to x* = [ 0.1997895  -0.2000151   0.20006513]
The accuracy is 0.0006
```

Problem 14 (Conjugate gradient method; 6 pt)**.** Consider the quadratic function

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^\top G\mathbf{x} - \mathbf{b}^\top\mathbf{x}$$

in four variables $\mathbf{x} = (x_1, x_2, x_3, x_4)$, where

$$G = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{pmatrix}$$

and $\mathbf{b} = (1, 0, 2, \sqrt{5})^\top$. Apply the conjugate gradient method to this problem with $\mathbf{x}_0 = (0, 0, 0, 0)^\top$ and show that it converges in two iterations to the exact solution of $G\mathbf{x} = \mathbf{b}$.

Solution to the problem 14. The provided code with its output is below:

```
In [1]: import numpy as np
        from scipy import linalg as LA
        from math import sqrt

        class GCM():
            def __init__(self, G, b, eps=10e-5):
                self.G = G
                self.b = b
                self.eps = eps

            def gF(self, x):
                return self.G.dot(x) - self.b

            def alpha(self, p):
                return (p.T.dot(self.b) / p.T.dot(self.G.dot(p))).flatten()[0]

            def p_next(self, P, r):
                coefs = (P.T.dot(self.G.dot(r)).T / np.diag(P.T.dot(G.dot(P)))).flatten()
                p = r - np.sum(coefs * P, 1, keepdims=True)
                return p

            def run(self, x0):
                print('\tx_0 = {}'.format(x0.flatten()))
                print('iteration 1:')
                p0 = -self.gF(x0)
                P = p0
                alpha0 = self.alpha(p0)
                x_prev = x0
                x_next = x_prev + alpha0 * p0
                norm_res = LA.norm(self.G.dot(x_next) - self.b)
                print('\tp_0 = {}'.format(p0.flatten()))
                print('\tx_1 = {}'.format(x_next.flatten()))
```

```
                print('\t||Gx-b|| = {:.3f}'.format(norm_res))
                for i in range(1,self.G.shape[0]):
                    print('iteration {}:'.format(i+1))
                    x_prev = x_next
                    r_next = -self.gF(x_prev)
                    p_next = self.p_next(P, r_next)
                    P = np.hstack((P, p_next))
                    alpha_next = self.alpha(p_next)
                    x_next = x_prev + alpha_next * p_next
                    norm_res = LA.norm(G.dot(x_next) - b)
                    print('\tp_{} = {}'.format(i, p_next.flatten()))
                    print('\tx_{} = {}'.format(i+1, x_next.flatten()))
                    print('\t||Gx-b|| = {:.3f}'.format(norm_res))
                    if norm_res < self.eps:
                        self.x = x_next
                        self.P = P
                        print('\nThe method converged to x* = {}'.format(self.x.flatten()))
                        break

        G = np.array([
        [2, -1, 0, 0],
        [-1, 2, -1, 0],
        [0, -1, 2, -1 ],
        [0,  0, -1, 2],
        ])
        b = np.array([[1,  0, 2, sqrt(5)]]).T
        x0 = np.zeros((4,1))
        GCM(G, b).run(x0)


        x_0 = [0. 0. 0. 0.]
iteration 1:
        p_0 = [ 1.         -0.          2.          2.23606798]
        x_1 = [0.9045085  0.         1.80901699 2.02254249]
        ||Gx-b|| = 2.860
iteration 2:
        p_1 = [0.00911863 2.71352549 2.04077974 1.82940686]
        x_2 = [0.91119459 1.98965125 3.30538767 3.36392721]
        ||Gx-b|| = 1.838
iteration 3:
        p_2 = [ 1.17102717  1.35770074  1.58544583 -0.43103244]
        x_3 = [1.60398844 2.79288344 4.2433566  3.10892321]
        ||Gx-b|| = 0.906
iteration 4:
        p_3 = [ 0.86953519  0.59157947 -0.19954978  0.15681188]
        x_4 = [2.0472136  3.09442719 4.14164079 3.18885438]
        ||Gx-b|| = 0.000
```

```
The method converged to x* = [2.0472136  3.09442719 4.14164079 3.18885438]
```

So, the GCM convergd to the exact solution $G\mathbf{x} = \mathbf{b}$ in $n = 4$ steps, as it has to.