Homework no. 5

Let $p \in \mathbf{N}^*$ and $n \in \mathbf{N}^*$ be the size of matrix $A \in \mathbf{R}^{p \times n}$, ϵ - computation error, matrix $A \in \mathbf{R}^{p \times n}$.

- For $p = n$, approximately compute the eigenvalues and eigenvectors for the symmetric matrix A ($A = A^T$) using Jacobi's method.

  Verify that:

  $$A^{init}U \approx U\Lambda \quad , \quad U = [u^1 \ u^2 \ \cdots \ u^n] \quad , \quad \Lambda = \text{diag}[\lambda_1, \ \lambda_2, \ \cdots \ \lambda_n]$$

  where λ_i are the approximate eigenvalues, u^i are the corresponding eigenvectors and A^{init} is a copy of the initial matrix.

  The verification $A^{init}U \approx \Lambda U$ is done by displaying the following norm:

  $$||A^{init}U - U\Lambda||.$$

- Compute the following sequence of matrices, $A^{(k)}$:

  $$A^{(0)} = A = L^0(L^0)^T, A^{(1)} = (L^0)^T L^0, A^{(1)} = L^1(L^1)^T, A^{(2)} = (L^1)^T L^1, ...$$

  $$A^{(k)} = L^k (L^k)^T (\text{ the Cholesky factorization of matrix } A^{(k)}) \ ,$$
  $$A^{(k+1)} = (L^k)^T L^k$$

  One stops the computations when the difference between two consecutive matrices is sufficiently small ($||A^{(k)} - A^{(k-1)}|| < \epsilon$) or when a pre-established number of iterations were performed ($k > k_{max}$). One doesn't need to memorize all the matrices from the sequence. Display the last computed matrix. What is its form? What information can be found in this matrix? For computing the Cholesky decomposition use the function implemented for **Homework 2** or a function from the numerical library that computes LU decompositions.

- For $p > n$, using the **S**ingular **V**alue **D**ecomposition) implemented in the library from **Homework 2**, compute and display:
  - the singular values of matrix A,
  - the rank of matrix A,

– the conditioning number of matrix A,

– Moore-Penrose pseudo-inverse of matrix A, $A^I \in \mathbf{R}^{n \times p}$,

$$A^I = VSU^T$$

– compute the least squares pseudo-inverse:

$$A^J = (A^T * A)^{-1} * A^T$$

and display the norm:

$$\|A^I - A^J\|_1$$

For the rank and the conditioning number of the matrix implement the formulae described in this document and also use the functions from the library that compute these values.

Bonus 20 pt.: Use for storing matrix A a vector v of size $\frac{n(n+1)}{2}$ (one stores only the elements from the lower triangular part of matrix A, the other can be addressed using the symmetry property). Adapt Jacobi's algorithm for this type of storage for matrix A.

Eigenvectors and eigenvalues - definitions

Consider $A \in \mathbf{R}^{n \times n}$ a square, real matrix of size n. An *eigenvalue* for matrix A, is a complex number $\lambda \in \mathbf{C}$, for which there exists a non-zero vector $u \neq 0$, called şi *eigenvector* associated to eigenvalue λ such that:

$$Au = \lambda\, u$$

The eigenvalues of matrix A can also be defined as the roots of the characteristic polynomial for matrix A, $p_A(\lambda)$:

$$p_A(\lambda) \;=\; \det(\lambda I - A) = 0$$

The characteristic polynomial has degree n and real coefficients, thus any matrix of size n has n eigenvalues (real and/or complex conjugated).

The symmetric matrices have only real eigenvalues.

Jacobi's method for approximating the eigenvalues of symmetric matrices

For approximating the eigenvalues of a matrix one can employ the similarity transformation. Two matrices are considered *similar* ($A \sim B$) if there exists a non-singular matrix P such that $A = PBP^{-1}$ ($\longleftrightarrow B = P^{-1}AP$). Note that if $A \sim B$ then $B \sim A$ is also true. The similarity transformations are used in the algorithms for numerically solving the eigenvalue-eigenvector problem because matrices that are similar have the same characteristic polynomial($p_A(\lambda) \equiv p_B(\lambda)$) and by consequence have the same eigenvalues.

Let $A \in \mathbf{R}^{n\times n}$ be a symmetric matrix ($A = A^T$). This type of matrices have only real eigenvalues.

The idea behind Jacobi's algorithm is to compute a sequence of matrices that are all similar with the initial matrix A. This sequence converges to a diagonal matrix.This diagonal matrix is similar with matrix A, thus the eigenvalues that we are looking for are the values from the diagonal of the limit matrix.

Computing the sequence of matrices

A rotation matrix $R_{pq}(\theta) = R_{pq} = (r_{ij})_{i,j=1,n}$ is a matrix that has the following form:

$$R_{pq}(\theta) = R_{pq} = \begin{pmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & & & & & & \\ 0 & 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & & & & & & \\ 0 & 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & & & & & & \\ 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}$$

$$r_{ij} = \begin{cases} 1 & \text{for } i = j, \quad i \neq p \text{ and } i \neq q \\ c & \text{for } i = j, \quad i = p \text{ or } i = q \\ s & \text{for } i = p, \ j = q \\ -s & \text{for } i = q, \ j = p \\ 0 & \text{otherwise} \end{cases}$$

where $p, q \in \{1, \ldots, n\}$ are indices, c and s are two real numbers that satisfy the relation $c^2 + s^2 = 1$ (c and s can be chosen such that $c = \cos\theta, s = \sin\theta$).

The sequence of matrices $\{A^{(k)}\} \subseteq \mathbf{R}^{n \times n}$ is calculated in the following way:

$$A^{(0)} = A \quad , \quad A^{(k+1)} = R_{pq}(\theta) A^{(k)} R_{pq}^T(\theta)$$

where $R_{pq}(\theta)$ are rotation matrices.

- **indices** (p, q) are selected to be the indices of the greatest non-diagonal element (in absolute value) of the current matrix:

$$\begin{aligned} |a_{pq}^{(k)}| &= \max\{|a_{ij}^{(k)}|;\ i = 1, \ldots, n\ ,\ j = 1, \ldots, n\ ,\ i \neq j\} = (A = A^T) = \\ &= \max\{|a_{ij}^{(k)}|;\ i = 2, \ldots, n\ ,\ j = 1, \ldots, i-1\} \end{aligned} \tag{1}$$

  (because the matrices $A^{(k)}$ are all symmetric, one can search the element $a_{pq}^{(k)}$ only in the strictly lower triangular part of the matrix $A^{(k)}$)

- **the angle** θ **(**$c = \cos\theta,\ s = \sin\theta,\ t = \mathrm{tg}\theta$**)** is selected such that the elements in positions (p, q) and (q, p) on matrix $A^{(k+1)}$ are zero, i.e.,

$$a_{pq}^{(k+1)} = a_{qp}^{(k+1)} = 0.$$

Jacobi's Algorithm

$k = 0;\ U = I_n;$
compute indices p and q (use (1)) ;
compute the angle θ, that is c, s and t;
while ($A \neq$ diagonal matrix and $k \leq k_{max}$)
{
$A = R_{pq}(\theta)\, A\, R_{pq}^T(\theta)$;
(see below presented formulae (5))
$U = U\, R_{pq}^T(\theta)$;
(use below presented formulae (7))
compute indices p and q (use (1));
compute the angle θ, that is c, s and t;
(use below presented formulae (3) and (4))
$k = k + 1;$
}

At the end of this algorithm (if $k < k_{max}$) one obtains in matrix $A = A^{final}$ a (approximately) diagonal matrix, the values from the diagonal are approximations of the eigenvalues of matrix A, and the columns of matrix U (orthogonal matrix) are approximations of the corresponding eigenvectors.

$$A^{final} = U^T A^{init} U$$

Step k of the algorithm

On computes in this step of the algorithm the matrix B starting from matrix A using the relation:

$$B = R_{pq}(\theta)\ A\ R_{pq}^T(\theta)$$

and also one computes matrix V starting from matrix U:

$$V = U\ R_{pq}^T(\theta).$$

Computing matrix B from matrix A is performed using the following formulae:

$$\left\{\begin{array}{lcl} b_{pj} & = & b_{jp} = c\ a_{pj} + s\ a_{qj}\ ,\ j = 1, 2, \ldots, n, \quad j \neq p,\ j \neq q \\ b_{qj} & = & b_{jq} = -s\ a_{pj} + c\ a_{qj}\ ,\ j = 1, 2 \ldots, n, \quad j \neq p,\ j \neq q \\ b_{pp} & = & c^2\, a_{pp} + s^2\, a_{qq} + 2\, c\, s\, a_{pq} \\ b_{qq} & = & s^2\, a_{pp} + c^2\, a_{qq} - 2\, c\, s\, a_{pq} \\ b_{pq} & = & b_{qp} = (c^2 - s^2)\, a_{pq} + c\, s\, (a_{qq} - a_{pp}) \\ b_{ij} & = & a_{ij} \quad \hat{\text{i}}\text{n rest} \end{array}\right. \tag{2}$$

One finds angle θ by setting to zero the elements $b_{pq} = b_{qp} = 0$, that is:

$$(c^2 - s^2)a_{pq} + c\,s\,(a_{qq} - a_{pp}) = 0.$$

From the avove relation we deduce that:

$$\alpha = \mathrm{cotg}(2\theta) = \frac{(a_{pp} - a_{qq})}{2a_{pq}}$$

Note by $t = \mathrm{tg}\theta$. We obtain:

$$\mathrm{cotg}(2\theta) = \frac{(1 - t^2)}{2t}$$

from where we deduce that t satisfies the equation:

$$t^2 + 2\alpha\, t - 1 = 0$$

thus

$$t = -\alpha + (\alpha^2 + 1)^{1/2} \text{ sau } t = -\alpha - (\alpha^2 + 1)^{1/2}.$$

Between the two possible values for t one chooses the root with minimal absolute value($\theta \in [0, \pi/4]$):

$$t = -\alpha + \mathrm{sign}(\alpha)\sqrt{\alpha^2 + 1} = \begin{cases} -\alpha + \sqrt{\alpha^2 + 1} & \text{if } \alpha \geq 0 \\ \\ -\alpha - \sqrt{\alpha^2 + 1} & \text{if } \alpha < 0 \end{cases} \tag{3}$$

$$\mathrm{sign}(\alpha) = \begin{cases} 1 & \text{if } \alpha \geq 0 \\ -1 & \text{if } \alpha < 0 \end{cases}$$

One gets:

$$c = \frac{1}{\sqrt{1 + t^2}} \quad , \quad s = \frac{t}{\sqrt{1 + t^2}} \tag{4}$$

Case $a_{pq} = 0$

From the fact that a_{pq} is the greatest non-diagonal element (in absolute value), the situation when $a_{pq} = 0$ signals that the current computed matrix A is a diagonal matrix. One stops the algorithm, the goal is achieved - the diagonal elements of current matrix A contains approximations of the sought eigenvalues. Thus, the test:

$$A \neq \text{ matrice diagonală}$$

from the above described algorithm can be replaced wit the test:

$$|a_{pq}| > \epsilon$$

where ϵ is the computation error.
Note that :

$$\begin{aligned} b_{pp} - a_{pp} &= s^2(a_{qq} - a_{pp}) + 2c\ s\ a_{pq} = 2s\ (c - \alpha\ s)a_{pq} = \\ &= 2s\ [c - (c^2 - s^2)s/(2c\ s)]a_{pq} = t\ a_{pq} \end{aligned}$$

In a similar way one deduces that:

$$b_{qq} - a_{qq} = -t\ a_{pq}$$

In step k the computations $A = R_{pq}(\theta)\, A\, R_{pq}^T(\theta)$ can be performed without using a auxiliary matrix B, in the following way:

$$\begin{aligned} a_{pj} &= c\ a_{pj} + s\ a_{qj}\ , \quad j = 1, 2, \ldots, n\,,\ j \neq p\,,\ j \neq q, \\ a_{qj} &= a_{jq} = -s\ a_{jp} + c\ a_{qj}\ , \quad j = 1, 2 \ldots, n\,,\ j \neq p\,, j \neq q, \\ a_{jp} &= a_{pj}\ , \quad j = 1, 2 \ldots, n\,,\ j \neq p\,, j \neq q, \\ a_{pp} &= a_{pp} + t\ a_{pq} \\ a_{qq} &= a_{qq} - t\ a_{pq} \\ a_{pq} &= a_{qp} = 0 \end{aligned} \tag{5}$$

Computing matrix V from matrix U is performed by changing only the columns p and q in the following way:

$$\left\{ \begin{array}{rcl} v_{ip} &=& c\ u_{ip} + s\ u_{iq}\ ,\ i = 1, 2, \ldots, n, \\ v_{iq} &=& -s\ u_{ip} + c\ u_{iq}\ ,\ i = 1, 2 \ldots, n. \end{array} \right. \tag{6}$$

The computations can be performed directly in matrix U, without using matrix V:

$$\left\{ \begin{array}{rcl} u_{ip} & = & c\ u_{ip} + s\ u_{iq}\ ,\ i = 1, 2, \ldots, n, \\ u_{iq} & = & -s\ u_{ip}^{old} + c\ u_{iq}\ ,\ i = 1, 2 \ldots, n. \end{array} \right. \tag{7}$$

Remark: Because matrix A is symmetric, one can store this matrix using a vector v size $\frac{n(n+1)}{2}$. One stores only the elements from the lower triangular part of the matrix. Vector v will contain the elements:

$$v : a_{11}, a_{21}, a_{22}, \ldots, a_{r1}, a_{r2}, \ldots, a_{rr}, \ldots, a_{n1}, a_{n2}, \ldots, a_{nn}$$

$$v_1 = a_{11}, v_2 = a_{21}, \cdots, v_{\frac{n(n+1}{2}} = a_{nn}$$

the other elements from matrix A can be retrieved using the symmetry property:

$$a_{ij} = a_{ji}$$

The problem that must be solved in order to implement Jacobi's algorithm with this type of matrix storage is the following:

For any given indices $i, j \in \{1, 2, \ldots, n\}$ (indices of elements from matrix A) find the index $k(i, j) \in \{1, 2, \ldots, \frac{n(n+1)}{2}\}$ (index of an element in vector v) such that:

$$a_{ij} = v_{k(i,j)} \qquad (a[i][j] = v[k(i, j)])$$

Singular Value Decomposition

Let $A \in \mathbf{R}^{p\times n}$. The Singular Value Decomposition for matrix A is given by the following relation:

$$A = USV^T \quad , \quad U \in \mathbf{R}^{p\times p} \ , \ S \in \mathbf{R}^{p\times n} \ , \ V \in \mathbf{R}^{n\times n}$$

with $U = [u_1 \ u_2 \ ... \ u_p]$ (the u_i vectors are the columns of matrix U) and $V = [v_1 \ v_2 \ ... \ v_n]$ orthogonal matrices and S a diagonal matrix:

$$\text{for } p \leq n \qquad S = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & \cdots & 0 \\ \vdots & & & & & \\ 0 & 0 & \cdots & \sigma_p & \cdots & 0 \end{pmatrix} \in \mathbf{R}^{p\times n}$$

$$\text{for } p > n \qquad S = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & \sigma_n \\ \vdots & & & \\ 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathbf{R}^{p\times n}$$

where the nonnegative numbers $\sigma_i \geq 0, \forall i$ are the singular values of matrix A.

The rank of matrix A can be computed as the number of strictly positive singular values:

$$\text{rang } (A) = \text{ number of singular values } \sigma_i > 0.$$

The conditioning number of matrix A is the ratio between the largest singular value and the smallest strictly positive singular value: strict pozitivă.

$$k_2(A) = \frac{\sigma_{\max}}{\sigma_{\min}} \quad ,$$

$$\sigma_{\max} = \max\{\sigma_i; \sigma_i \text{ singular value } \} \ ,$$

$$\sigma_{\min} = \min\{\sigma_i; \sigma_i > 0 \text{ singular value } \}$$

The Moore-Penrose pseudo-inverse of matrix A is computed using the formula:

$$A^I = VS^IU^T.$$

The matrix S^I is computed using the formula below displayed. Assume that we computed for matrix $A \in \mathbf{R}^{p\times n}$ the singular value decomposition. Let $\sigma_1, \sigma_2, \ldots, \sigma_r > 0$ be the strictly positive singulr values for matrix A, $r = \text{rang}(A)$.

Matrix $S^I \in \mathbf{R}^{n\times p}$ is computed using the relation:

$$S^I = \begin{pmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & \cdots & 0 & \cdots & 0 \\ \vdots & & & & & \\ 0 & 0 & \cdots & \frac{1}{\sigma_r} & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & & & & \\ 0 & 0 & \cdots & 0 & \cdots & 0 \end{pmatrix} \in \mathbf{R}^{n\times p}.$$

The vector $x^I = VS^IU^Tb$ can be considered the solution of the linear system $Ax = b$ even for non-square matrices (when $p \neq n$) and the system does not have a classical solution. When $p = n$ and the matrix A is non-singular the vector x^I is the same with the classical solution of system $Ax = b$.

Examples

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \text{ has the eigenvalues } \lambda_1 = -1\,,\ \lambda_2 = 0\,,\ \lambda_3 = 2$$

$$\lambda_1 = -1,\ u^1 = \begin{pmatrix} a \\ a \\ -a \end{pmatrix}\ a \in \mathbf{R}\ a \neq 0\ ,\quad \lambda_2 = 0,\ u^2 = \begin{pmatrix} b \\ -b \\ 0 \end{pmatrix}\ b \in \mathbf{R}\ b \neq 0$$

$$\lambda_3 = 2 \quad u^3 = \begin{pmatrix} c \\ c \\ 2c \end{pmatrix}\ c \in \mathbf{R}\ c \neq 0$$

$A = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 1 & 2 \\ 2 & 2 & 2 \end{pmatrix}$ has the eigenvalues $\lambda_1 = 0\,,\ \lambda_2 = 2(1-\sqrt{2})\,,\ \lambda_3 = 2(1+\sqrt{2})$

$2(1-\sqrt{2}) \approx -0.828427,\ 2(1+\sqrt{2}) \approx 4.828437$

$$\lambda_1 = 0 \quad u^1 = \begin{pmatrix} a \\ -a \\ 0 \end{pmatrix}\ a \in \mathbf{R}\ a \neq 0$$

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \text{ has the eigenvalues } \lambda_1 = \lambda_2 = 0\,,\ \lambda_3 = \lambda_4 = 2$$

$$\lambda_{1,2} = 0 \quad u^{1,2} = \begin{pmatrix} a \\ b \\ -a \\ -b \end{pmatrix}\ a, b \in \mathbf{R} \quad , \quad \lambda_{3,4} = 2 \quad u^{3,4} = \begin{pmatrix} c \\ d \\ c \\ d \end{pmatrix}\ c, d \in \mathbf{R}$$

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{pmatrix} \text{ has the eigenvalues } \lambda_1 = \lambda_2 = 0\,,\ \lambda_{3,4} = 2(4 \pm \sqrt{21})$$

$2(4+\sqrt{21}) \approx 17.165151\ ,\ 2(4-\sqrt{21}) \approx -1.165151$

$$\lambda_{1,2} = 0 \quad u^{1,2} = \begin{pmatrix} a \\ b \\ -3a-2b \\ 2a+b \end{pmatrix}\ a, b \in \mathbf{R}$$