ETH Lecture 401-0663-00L Numerical Methods for CSE

Mid-term Exam

Autumn Term 2020

Oct 31, 2020, 10:00, HG E1.2 / HG E7 / HG F1

Don't panic!

Family Name		%
First Name		
Department		
Legi Nr.		
Date	Oct 31, 2020	

Points:

	1	2	3	Total
max	10	9	8	
achvd				

- This is a **closed-book exam.**
- Keep only writing material and your ETH ID card on the table.
- Keep mobile phones, tablets, smartwatches, etc. turned off in your bag.
- Fill in this cover sheet first.
- Turn the cover sheet only when instructed to do so.
- Then write your name and ETH ID number on every page.
- **Write your answers in the appropriate fields on these problem sheets.**
- **Wrong ticks in multiple-choice boxes will lead to points being subtracted.** Hence, mere guessing is really dangerous! If you have no clue, leave all tickboxes empty.
- If you change your mind about an answer to an MC-question, write a clear NO next to the old answer, draw fresh tickboxes and fill them.
- **Anything written outside the answer boxes will not be taken into account.**
- Do not write with red/green color or with pencil.
- Make sure to hand in every sheet.
- Two blank pages handed out with the exam: space for notes
- **Duration: 30 minutes.**

Throughout the exam use the notations introduced in class, in particular [Lecture → Section 1.1.1]:

- $(\mathbf{A})_{i,j}$ to refer to the entry of the matrix $\mathbf{A} \in \mathbb{K}^{m,n}$ at position (i, j).

- $(\mathbf{A})_{:,i}$ to designate the ith-column of the matrix $\mathbf{A}$,
- $(\mathbf{A})_{i,:}$ to denote the i-th row of the matrix $\mathbf{A}$,
- $(\mathbf{A})_{i:j,k:\ell}$ to single out the sub-matrix $\left[(\mathbf{A})_{r,s}\right]_{\substack{i\leq r\leq j\\ k\leq s\leq \ell}}$ of the matrix $\mathbf{A}$,
- $(\mathbf{x})_k$ to reference the k-th entry of the vector $\mathbf{x}$,
- $\mathbf{e}_j \in \mathbb{R}^n$ to write the j-th Cartesian coordinate vector,
- $\mathbf{I}$ to denote the identity matrix,
- $\mathbf{O}$ to write a zero matrix,
- $\mathcal{P}_n$ for the space of (univariate polynomials of degree $\leq n$),
- and superscript indices in brackets to denote iterates: $\mathbf{x}^{(k)}$, etc.

By default, vectors are regarded as column vectors.

Problem 0-1: Economical QR-decomposition

The economical QR-decomposition is a fundamental matrix factorization in numerical linear algebra. This problem reviews some of its properties.

This problems is based on the contents of [Lecture → Section 3.3.3].

Given a matrix $\mathbf{A} \in \mathbb{R}^{m,n}$, $m > n$, $\mathbf{A} \neq \mathbf{O}$, we write $\mathbf{A} = \mathbf{QR}$ for its **economical (thin) QR decomposition**.

(0-1.a) (2 pts.) Fill in the correct matrix sizes

$$\mathbf{Q} \in \mathbb{R}^{\square,\square} \quad , \quad \mathbf{R} \in \mathbb{R}^{\square,\square} \; .$$

SOLUTION of (0-1.a):

$$\mathbf{Q} \in \mathbb{R}^{m,n} \quad , \quad \mathbf{R} \in \mathbb{R}^{n,n} \; .$$

▲

(0-1.b) (4 pts.) Decide, whether the following statements are true for any $\mathbf{A} \in \mathbb{R}^{m,n} \setminus \{\mathbf{O}\}$, $m > n$.

1. The matrix $\mathbf{Q}$ is an orthogonal matrix.

   ◯ YES ◯ NO

2. The columns of $\mathbf{Q}$ provide an orthonormal basis of $\mathcal{R}(\mathbf{A})$.

   ◯ YES ◯ NO

3. $\operatorname{rank}(\mathbf{R}) = n$.

   ◯ YES ◯ NO

4. $\operatorname{rank}(\mathbf{Q}) = n$.

   ◯ YES ◯ NO

SOLUTION of (0-1.b):

1. NO: because $\mathbf{Q}$ is not even square in general.

2. NO: Consider a rank-1 matrix $\mathbf{A}$ with $n \geq 2$. Then $\mathbf{Q}$ has more columns than $\dim \mathcal{R}(\mathbf{A})$. As a consequence, the columns of $\mathbf{Q}$ cannot be a basis of $\mathcal{R}(\mathbf{A})$.
3. NO: because $\operatorname{rank}(\mathbf{R}) = \operatorname{rank}(\mathbf{A})$, but $\operatorname{rank}(\mathbf{A}) < n$ is not excluded.
4. YES: because the n orthonormal columns of $\mathbf{Q} \in \mathbb{R}^{m,n}$ are linearly independent.

▲

(0-1.c) ⚁ (2 pts.) How many left-multiplications by Householder matrices does it take to compute $\mathbf{R}$?

○ mn	○ n	○ $\frac{1}{2}n(n-1)$	○ $m-1$

SOLUTION of (0-1.c):

We need n left-multiplications with Householder matrices in order to annihilate the bottom entries of the n columns of $\mathbf{A}$.

Note that $m > n$ was assumed. In the case $m \leq n$ the computation of $\mathbf{R}$ requires only $m-1$ left-multiplications with Householder matrices.

▲

(0-1.d) ⚁ (2 pts.) What is the asymptotic complexity of EIGEN's `HouseholderQR()` methods applied to dense $m \times n$ matrices, $m > n$, for $m, n \to \infty$?

○ $O(mn)$	○ $O(m^2 n)$	○ $O(m(m+n^2))$	○ $O(mn^2)$

SOLUTION of (0-1.d):

$O(mn^2)$ according to [Lecture → § 3.3.3.37].

▲

End Problem 0-1, 10 pts.

Problem 0-2: Evaluation of a quadratic form in COO format

This problem examines the algorithmic realization of the evaluation of a quadratic form described by a sparse matrix in triplet/COO storage format.

Familiarity with C++/EIGEN and [Lecture → Section 2.7.1] is required.

In an EIGEN-based numerical code a sparse matrix in COO format is represented by the data type

```
using COOMat = std::vector<Eigen::Triplet<double>>;
```

This function converts a matrix in triplet/COO format into a dense matrix format:

```
Eigen::MatrixXd cooToDense(const COOMat &A) {
  Eigen::Index m = 0;
  Eigen::Index n = 0;
  for (const Eigen::Triplet<double> &t : A) {
    m = (m > t.row()) ? m : t.row() + 1;
    n = (n > t.row()) ? n : t.col() + 1;
  }
  Eigen::MatrixXd M = Eigen::MatrixXd::Zero(m, n);
  for (const Eigen::Triplet<double> &t : A) {
    M(t.row(), t.col()) += t.value();
  }
  return M;
}
```

(0-2.a) (9 pts.) The following function is supposed to compute $(\mathbf{1}-\mathbf{x})^\top \mathbf{A}\mathbf{x}$, $\mathbf{1} \in \mathbb{R}^n$ a vector with components all $= 1$, for a large sparse matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ given in triplet/COO format and a vector $\mathbf{x} \in \mathbb{R}^n$:

```
double evalQuadFormCOO(const COOMat &A, const Eigen::VectorXd &x) {
  double s = 0.0;
  for (const Eigen::Triplet<double> &t : A) {
    assert((t.row() [    ] x.size()) && (t.col() [    ] x.size()));
    s [  ] ([                    ]) * [              ] * [              ];
  }
  return s;
}
```

Supplement the missing parts of C++ code in the boxes.

SOLUTION of (0-2.a):

C++ code 0.2.1: Implementation of `evalQuadFormCOO()`

```
double evalQuadFormCOO(const COOMat &A, const Eigen::VectorXd &x) {
  double s = 0.0;
  for (const Eigen::Triplet<double> &t : A) {
    assert((t.row() < x.size()) && (t.col() < x.size()));
    s += (1.0 - x[t.row()]) * x[t.col()] * t.value();
  }
  return s;
```

```
}
```

▲

End Problem 0-2 , 9 pts.

Problem 0-3: Analysis of an EIGEN-based C++ code

In this problem you are asked to analyze a complex code and determine the computational cost of certain operations and the size of matrices stored in certain variables.

This problem assumes familiarity with EIGEN.

In a project you are expected to understand the following undocumented C++ function, which takes two arguments, a matrix $\mathbf{A} \in \mathbb{R}^{m,n}$, $m \geq n$, and an integer $d < n$.

C++ code 0.3.1: A complex EIGEN-based C++ code

```
2  std::pair<Eigen::VectorXd, Eigen::VectorXd> clsq2(const Eigen::MatrixXd &A,
3                                                     const unsigned d) {
4    unsigned int n = A.cols(), m = A.rows();
5    assert(m >= n);
6    const Eigen::MatrixXd R =
7      A.householderQr().matrixQR().template triangularView<Eigen::Upper>(); //
8    const auto V = R.block(d, d, m-d , n-d) //
9                    .jacobiSvd(Eigen::ComputeFullV) //
10                   .matrixV(); //
11   const auto y = V.col(n-d - 1); //
12   const auto b = R.block(0, d, d, n-d) * y; //
13   const auto S = R.topLeftCorner(d, d); //
14   const Eigen::VectorXd D = S.diagonal().cwiseAbs(); //
15   if (D.minCoeff() < (numeric_limits<double>::epsilon() * D.maxCoeff()) //
16     throw runtime_error("Upper left block of R not regular"); //
17   const auto z = -(S.template triangularView<Eigen::Upper>()).solve(b); //
18   return {z, y};
19 }
```

(0-3.a) ⚂ (4 pts.) Find out the type of matrix/vector stored in various variables

(i) `R` defined in Line 7 stores matrix/vector $\in \mathbb{R}^{\square,\square}$

(ii) `V` defined in Line 8 stores matrix/vector $\in \mathbb{R}^{\square,\square}$

(iii) `b` defined in Line 12 stores matrix/vector $\in \mathbb{R}^{\square,\square}$

(iv) `z` defined in Line 17 stores matrix/vector $\in \mathbb{R}^{\square,\square}$

HINT 1 for (0-3.a): The method `block(int i,int j,int p,int q)` of a matrix type in EIGEN returns a reference to the submatrix with upper left corner at position (i, j) and size $p \times q$. ⌟

SOLUTION of (0-3.a):

(i) `R` is the upper triangular part of the $\mathbf{R}$-factor of the QR-decomposition of the matrix $\mathbf{A} \in \mathbb{R}^{m,n}$. It has the same size as $\mathbf{R}$ and stores an $m \times n$-matrix.

(ii) `v` is the $\mathbf{V}$-factor of the SVD of a $(m-d)\times(n-d)$-matrix, hence of size $(n-d)\times(n-d)$.

(iii) `b` stores a column vector of length d obtained by the multiplication of a $d\times(n-d)$-matrix (in `V`) with and $n-d$-vector (in `y`).

(iv) `S` contains a $d\times d$-matrix $\mathbf{S}\in\mathbb{R}^{d,d}$. Hence `z` holds the vector $\mathbf{S}^{-1}\mathbf{b}\in\mathbb{R}^{d}$.

▲

(0-3.b) (4 pts.) Assuming that d is small and fixed, determine the asymptotic complexity in terms of $m,n\to\infty$ of the statements in different lines of Code 0.3.1

(i)	Line 7	Cost = $O($ $)$
(ii)	Line 8–Line 10	Cost = $O($ $)$
(iii)	Line 14	Cost = $O($ $)$
(iv)	Line 17	Cost = $O($ $)$

SOLUTION of (0-3.b):

(i) Line 7: Computation of the QR-decomposition of $\mathbf{A}\in\mathbb{R}^{m,n}$: cost = $O(mn^2)$, see [Lecture → § 3.3.3.37].

(ii) Line 8–Line 10: Computation of the SVD of an $(m-d)\times(n-d)$-matrix, $m\geq n$, which is an economical SVD, because the full $\mathbf{Q}$-factor is not requested: Cost = $O(mn^2)$, see [Lecture → § 3.4.2.2].

(iii) Line 14: Extraction of a d-vector by taking the absolute values of the entries of the diagonal of a $d\times d$-matrix. Cost = $O(1)$, because d was supposed to be fixed and small.

(iv) Line 17: Solution of a small $d\times d$ linear system of equations. Cost = $O(1)$, because d was supposed to be fixed and small.

▲

End Problem 0-3 , 8 pts.