

## Problem Sheet 2

You should try to your best to do the core problems. If time permits, please try to do the rest as well.

### Problem 1. Lyapunov Equation (core problem)

Any linear system of equations with a finite number of unknowns can be written in the “canonical form”  $\mathbf{Ax} = \mathbf{b}$  with a system matrix  $\mathbf{A}$  and a right hand side vector  $\mathbf{b}$ . However, the LSE may be given in a different form and it may not be obvious how to extract the system matrix. This task gives an intriguing example and also presents an important *matrix equation*, the so-called [Lyapunov Equation](#).

Given  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , consider the equation

$$\mathbf{AX} + \mathbf{XA}^T = \mathbf{I} \quad (5)$$

with unknown  $\mathbf{X} \in \mathbb{R}^{n \times n}$ .

(1a)  $\square$  Show that for a fixed matrix  $\mathbf{A} \in \mathbb{R}^{n,n}$  the mapping

$$L : \begin{cases} \mathbb{R}^{n,n} & \rightarrow & \mathbb{R}^{n,n} \\ \mathbf{X} & \mapsto & \mathbf{AX} + \mathbf{XA}^T \end{cases}$$

is linear.

HINT: Recall from linear algebra the definition of a linear mapping between two vector spaces.

In the sequel let  $\text{vec}(\mathbf{M}) \in \mathbb{R}^{n^2}$  denote the column vector obtained by reinterpreting the internal coefficient array of a matrix  $M \in \mathbb{R}^{n,n}$  stored in column major format as the data array of a vector with  $n^2$  components. In MATLAB,  $\text{vec}(\mathbf{M})$  would be the column vector obtained by `reshape(M, n*n, 1)` or by `M(:)`. See [1, Rem. 1.2.18] for the implementation with Eigen.

Problem (5) is equivalent to a linear system of equations

$$\mathbf{C}\text{vec}(\mathbf{X}) = \mathbf{b} \quad (6)$$

with system matrix  $\mathbf{C} \in \mathbb{R}^{n^2, n^2}$  and right hand side vector  $\mathbf{b} \in \mathbb{R}^{n^2}$ .

**(1b)** ☐ Refresh yourself on the notion of “sparse matrix”, see [1, Section 1.7] and, in particular, [1, Notion 1.7.1], [1, Def. 1.7.3].

**(1c)** ☐ Determine  $\mathbf{C}$  and  $\mathbf{b}$  from (6) for  $n = 2$  and

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ -1 & 3 \end{bmatrix}.$$

**(1d)** ☐ Use the Kronecker product to find a general expression for  $\mathbf{C}$  in terms of a general  $\mathbf{A}$ .

**(1e)** ☐ Write a MATLAB function

```
function C = buildC (A)
```

that returns the matrix  $\mathbf{C}$  from (6) when given a square matrix  $\mathbf{A}$ . (The function `kron` may be used.)

**(1f)** ☐ Give an upper bound (as sharp as possible) for  $\text{nnz}(\mathbf{C})$  in terms of  $\text{nnz}(\mathbf{A})$ . Can  $\mathbf{C}$  be legitimately regarded as a sparse matrix for large  $n$  even if  $\mathbf{A}$  is dense?

HINT: Run the following MATLAB code:

```
n=4;  
A=sym('A',[n,n]);  
I=eye(n);  
C=kron(A,I)+kron(I,A)
```

**(1g)** ☐ Implement a C++ function

```
Eigen::SparseMatrix<double> buildC(const MatrixXd &A)
```

that builds the Eigen matrix  $\mathbf{C}$  from  $\mathbf{A}$ . Make sure that initialization is done efficiently using an intermediate triplet format. Read [1, Section 1.7.3] very carefully before starting.

(1h) ☐ Validate the correctness of your C++ implementation of `buildC` by comparing with the equivalent Matlab function for  $n = 5$  and

$$A = \begin{bmatrix} 10 & 2 & 3 & 4 & 5 \\ 6 & 20 & 8 & 9 & 1 \\ 1 & 2 & 30 & 4 & 5 \\ 6 & 7 & 8 & 20 & 0 \\ 1 & 2 & 3 & 4 & 10 \end{bmatrix}.$$

(1i) ☐ Write a C++ function

```
void solveLyapunov(const MatrixXd & A, MatrixXd & X)
```

that returns the solution of (5) in the  $n \times n$ -matrix  $\mathbf{X}$ , if  $A \in \mathbb{R}^{n,n}$ .

(1j) ☐ Test your C++ implementation of `solveLyapunov` by comparing with Matlab for the test case proposed in (1h).

## Problem 2. Partitioned Matrix (core problem)

Based on the block view of matrix multiplication presented in [1, § 1.3.13], we looked a *block elimination* for the solution of block partitioned linear systems of equations in [1, § 1.6.91]. Also of interest are [1, Rem. 1.6.45] and [1, Rem. 1.6.43] where LU-factorization is viewed from a block perspective. Closely related to this problem is [1, Ex. 1.6.94], which you should study again as warm-up to this problem.

Let the matrix  $\mathbf{A} \in \mathbb{R}^{n+1,n+1}$  be partitioned according to

$$\mathbf{A} = \begin{bmatrix} \mathbf{R} & \mathbf{v} \\ \mathbf{u}^T & 0 \end{bmatrix}, \quad (7)$$

where  $\mathbf{v} \in \mathbb{R}^n$ ,  $\mathbf{u} \in \mathbb{R}^n$ , and  $\mathbf{R} \in \mathbb{R}^{n \times n}$  is upper triangular and regular.

(2a) ☐ Give a necessary and sufficient condition for the triangular matrix  $\mathbf{R}$  to be invertible.

(2b) ☐ Determine expressions for the subvectors  $\mathbf{z} \in \mathbb{R}^n, \xi \in \mathbb{R}$  of the solution vector of the linear system of equations

$$\begin{bmatrix} \mathbf{R} & \mathbf{v} \\ \mathbf{u}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \xi \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \beta \end{bmatrix}$$

for arbitrary  $\mathbf{b} \in \mathbb{R}^n, \beta \in \mathbb{R}$ .

HINT: Use blockwise Gaussian elimination as presented in [1, § 1.6.91].

(2c)  Show that  $\mathbf{A}$  is regular if and only if  $\mathbf{u}^T \mathbf{R}^{-1} \mathbf{v} \neq 0$ .

(2d)  Implement the C++ function

```
template <class Matrix, class Vector>
void solvelse(const Matrix & R, const Vector & v, const
             Vector & u, const Vector & b, Vector & x);
```

for computing the solution of  $\mathbf{Ax} = \mathbf{b}$  (with  $\mathbf{A}$  as in (7)) efficiently. Perform size check on input matrices and vectors.

HINT: Use the decomposition from (2b).

HINT: you can rely on the `triangularView()` function to instruct EIGEN of the triangular structure of  $\mathbf{R}$ , see [1, Code 1.2.12].

HINT: using the construct:

```
typedef typename Matrix::Scalar Scalar;
```


you can obtain the scalar type of the `Matrix` type (e.g. `double` for `MatrixXd`). This can then be used as:

```
Scalar a = 5;
```


HINT: using `triangularView` and templates you may incur in weird compiling errors. If this happens to you, check <http://eigen.tuxfamily.org/dox/TopicTemplateKeyword.html>

HINT: sometimes the C++ keyword `auto` (only in std. C++11) can be used if you do not want to explicitly write the return type of a function, as in:

```
MatrixXd a;
auto b = 5*a;
```

(2e)  Test your implementation by comparing with a standard LU-solver provided by EIGEN.

HINT: Check the page [http://eigen.tuxfamily.org/dox/group\\_\\_TutorialLinearAlgebra.html](http://eigen.tuxfamily.org/dox/group__TutorialLinearAlgebra.html).

(2f)  What is the asymptotic complexity of your implementation of `solve()` in terms of problem size parameter  $n \rightarrow \infty$ ?


### Problem 3. Banded matrix

For  $n \in \mathbb{N}$  we consider the matrix

$$\mathbf{A} := \begin{bmatrix} 2 & a_1 & 0 & \dots & \dots & \dots & 0 \\ 0 & 2 & a_2 & 0 & \dots & \dots & 0 \\ b_1 & 0 & \ddots & \ddots & \ddots & & \vdots \\ 0 & b_2 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & a_{n-1} \\ 0 & 0 & \dots & 0 & b_{n-2} & 0 & 2 \end{bmatrix} \in \mathbb{R}^{n,n}$$


with  $a_i, b_i \in \mathbb{R}$ .

*Remark.* The matrix  $\mathbf{A}$  is an instance of a banded matrix, see [1, Section 1.7.6] and, in particular, the examples after [1, Def. 1.7.53]. However, you need not know any of the content of this section for solving this problem.

(3a)  Implement an *efficient* C++ function:


```
1  template <class Vector>
2  void multAx(const Vector & a, const Vector & b, const
    Vector & x, Vector & y);
```

for the computation of  $\mathbf{y} = \mathbf{A}\mathbf{x}$ .

(3b)  Show that  $\mathbf{A}$  is invertible if  $a_i, b_i \in [0, 1]$ .


HINT: Give an indirect proof that  $\ker \mathbf{A}$  is trivial, by looking at the largest (in modulus) component of an  $\mathbf{x} \in \ker \mathbf{A}$ .

*Remark.* That  $\mathbf{A}$  is invertible can immediately be concluded from the general fact that kernel vectors of irreducible, diagonally dominant matrices ( $\rightarrow$  [1, Def. 1.8.8]) must be multiples of  $[1, 1, \dots, 1]^\top$ . Actually, the proof recommended in the hint shows this fact first before bumping into a contradiction.

(3c)  Fix  $b_i = 0, \forall i = 1, \dots, n - 2$ . Implement an efficient C++ function

```
1  template <class Vector>
2  void solvelseAupper(const Vector & a, const Vector &
    r, Vector & x);
```


solving  $\mathbf{Ax} = \mathbf{r}$ .


(3d)  For general  $a_i, b_i \in [0, 1]$  devise an efficient C++ function:

```
1  template <class Vector>
2  void solvelseA(const Vector & a, const Vector & b,
    const Vector & r, Vector & x);
```

that computes the solution of  $\mathbf{Ax} = \mathbf{r}$  by means of Gaussian elimination. You cannot use any high level solver routines of EIGEN.

HINT: Thanks to the constraint  $a_i, b_i \in [0, 1]$ , pivoting is not required in order to ensure stability of Gaussian elimination. This is asserted in [1, Lemma 1.8.9], but you may just use this fact here. Thus, you can perform a straightforward Gaussian elimination from top to bottom as you have learned it in your linear algebra course.

(3e)  What is the asymptotic complexity of your implementation of `solvelseA` for  $n \rightarrow \infty$ .

(3f)  Implement `solvelseAEigen` as in (3d), this time using EIGEN's sparse elimination solver.




HINT: The standard way of initializing a sparse EIGEN-matrix efficiently, is via the triplet format as discussed in [1, Section 1.7.3]. You may also use direct initialization of a sparse matrix, provided that you `reserve()` enough space for the non-zero entries of each column, see [documentation](#).

## Problem 4. Sequential linear systems

This problem is about a sequence of linear systems, please see [1, Rem. 1.6.85]. The idea is that if we solve several linear systems with the same matrix  $A$ , that computational cost may be reduced by performing the LU decomposition only once.

Consider the following MATLAB function with input data  $A \in \mathbb{R}^{n,n}$  and  $b \in \mathbb{R}^n$ .

```
1 function X = solvepermb(A,b)
2 [n,m] = size(A);
3 if (n ~= numel(b)) || (m ~= numel(b)), error('Size
  mismatch'); end
4 X = [];
5 for l=1:n
6     X = [X,A\b];
7     b = [b(end);b(1:end-1)];
8 end
```

- (4a)  What is the asymptotic complexity of this function as  $n \rightarrow \infty$ ?
- (4b)  Port the MATLAB function `solvepermb` to C++ using EIGEN. (This means that the C++ code should perform exactly the same computations in exactly the same order.)
- (4c)  Design an efficient implementation of this function with asymptotic complexity  $O(n^3)$  in Eigen.

Issue date: 24.09.2015

Hand-in: 31.09.2015 (in the boxes in front of HG G 53/54).

Version compiled on: September 24, 2015 (v. 1.0).

## References

- [1] R. Hiptmair. *Lecture slides for course "Numerical Methods for CSE"*. <http://www.sam.math.ethz.ch/~hiptmair/tmp/NumCSE/NumCSE15.pdf>. 2015.