Dirk Praetorius
Michele Ruggeri

Summer semester 2019
June 7, 2019

Scientific programming in mathematics

Exercise sheet 11

References, keyword `const`, operator overloading, dynamic memory allocation in C++

Exercise 11.1. An upper triangular matrix $U \in \mathbb{R}^{n\times n}$ is a matrix such that $U_{jk} = 0$ for $k < j$, i.e.,

$$U = \begin{pmatrix} U_{00} & U_{01} & U_{02} & \cdots & U_{0,n-1} \\ & U_{11} & U_{12} & \cdots & U_{1,n-1} \\ & & U_{22} & \cdots & U_{2n-1} \\ & & & \ddots & \vdots \\ \mathbf{0} & & & & U_{n-1,n-1} \end{pmatrix}.$$

Write a class `UpperTriangularMatrix`, which stores the dimension $n \in \mathbb{N}$ (`int`) and the non-trivial coefficients U_{ij} in a dynamical vector $u \in \mathbb{R}^N$ (`double*`) of length $N = \frac{n(n+1)}{2} = \sum_{j=1}^n j$. The coefficients of U should be stored columnwise i.e., $U_{jk} = u_\ell$ for a suitable index ℓ, which depends on j and k. Derive a formula for $\ell = \ell(k,j)$. Implement the following features: Constructor (with optional initialization), destructor, copy constructor, and assignment operator. Test your implementation appropriately!

Exercise 11.2. Extend the class `UpperTriangularMatrix` from Exercise 11.1 by a method `int size() const` to read the dimension. Moreover, implement the capability of accessing the coefficients of the matrix via `( )`, i.e., for $0 \le j,k \le n-1$, the coefficient U_{jk} can be obtained by typing `U(j,k)`. Implement this feature for both `const` and non-`const` objects, i.e., in the class definition use the signatures

```
const double& operator()( int j, int k) const;
double& operator() (int j , int k);
```

Use `assert` to ensure that $0 \le j,k \le n-1$ and do not forget that $U_{jk} = 0$ if $k < j$. Test your implementation appropriately!

Exercise 11.3. Extend the class `UpperTriangularMatrix` from Exercise 11.1 with

- the possibility to print a matrix `U` to the screen via the syntax `cout << U`,
- a method `double columnSumNorm() const`, which computes and returns the column sum norm

$$\|U\|_1 = \max_{k=0,\ldots,n-1} \sum_{j=0}^{n-1} |U_{jk}|,$$

- and a method `double rowSumNorm() const`, which computes and returns the row sum norm

$$\|U\|_\infty = \max_{j=0,\ldots,n-1} \sum_{k=0}^{n-1} |U_{jk}|.$$

Note that the methods should access only the coefficients U_{jk} with $0 \le j \le k \le n-1$. Test your implementation appropriately!

Exercise 11.4. Overload the operator `+` for the class `UpperTriangularMatrix` from Exercise 11.1 to be able to compute the sum of two upper triangular matrices with matching dimensions. Use `assert` to ensure that the dimensions match. Test your implementation appropriately!

Exercise 11.5. Prove in a rigorous mathematical way using the formula for the matrix-matrix product

$$C_{ij} = \sum_{k=0}^{n-1} A_{ik} B_{kj} \quad \text{for } i, j = 0, \dots n-1$$

that the product $C = AB \in \mathbb{R}^{n \times n}$ of two upper triangular matrices $A, B \in \mathbb{R}^{n \times n}$ is an upper triangular matrix. Overload the operator `*` for the class `UpperTriangularMatrix` from Exercise 11.1 to be able to compute the product of two upper triangular matrices with matching dimensions. Check using `assert` that the matrices have the same dimension. Note that you need to compute only the coefficients C_{jk} for $0 \le j \le k \le n-1$ and the corresponding coefficients of the matrices A and B are the only ones that can be accessed. Test your implementation appropriately!

Exercise 11.6. Overload the operator `*` for the class `UpperTriangularMatrix` from Exercise 11.1 to be able to compute the matrix-vector product $b = Ux \in \mathbb{R}^n$ of an upper triangular matrix $U \in \mathbb{R}^{n \times n}$ and a vector $x \in \mathbb{R}^n$ with matching dimensions, i.e.,

$$b_j = \sum_{k=0}^{n-1} U_{jk} x_k \quad \text{for } j = 0, \dots, n-1.$$

To store vectors, use the class `Vector` from the lecture notes (slides 292–296). Use `assert` to ensure that the dimensions match. Note that the function can access only the coefficients U_{jk} with $0 \le j \le k \le n-1$. Test your implementation appropriately!

Exercise 11.7. Let $U \in \mathbb{R}^{n \times n}$ be an upper triangular matrix such that $U_{jj} \neq 0$ for all $j = 0, \dots, n-1$. Given $b \in \mathbb{R}^n$, there exists a unique $x \in \mathbb{R}^n$ such that $Ux = b$. Derive a formula to compute the solution $x \in \mathbb{R}^n$ of $Ux = b$ by using the formula for the matrix-vector product and the simplifications thereof which follow from the triangular structure of U. Overload the operator `|` so that typing `x = U | b` for an upper triangular matrix $U \in \mathbb{R}^{n \times n}$ (type `UpperTriangularMatrix` from Exercise 11.1) and a vector $b \in \mathbb{R}^n$ (type `Vector`, see slides 292–296 from the lecture notes) computes and returns the solution $x \in \mathbb{R}^n$ of the system $Ux = b$ (as object of type `Vector`). Use `assert` to check that the dimensions match and that $U_{jj} \neq 0$ for all j. Test your implementation appropriately!

Exercise 11.8. Determine the computational complexity of your implementations in Exercises 11.5, 11.6 and 11.7. If the operators have a runtime of 2 seconds for $n = 10^2$, which runtime do you expect for $n = 5 \cdot 10^3$? Justify your answers!