TP 1 : matrix data structures

Exercise 1 : vector operations

This exercise aims at defining enough operations between operands of type `std::vector<double>` that these can be considered decently modelling the algebra of $\mathbb{R}^n$ as a vector space.

- operator `+=` that adds two `std::vector<double>` and stores the result in the left hand side. Do a similar work with `-=`.
- operator `+` that adds two `std::vector<double>`. Do a similar work with the operator `-` for calculating the difference of two `std::vector<double>`.
- operator `,` that performs the scalar product of two `std::vector<double>`.
- a function `Norm` that computes the euclidean quadratic norm of a `std::vector`.
- operator `*=` that multiplies a `std::vector<double>` by a `double`.
- operator `*` that multiplies a `double` (left operand) by a `std::vector<double>` (right operand).

Exercise 2 : dense matrices

Write a class called `DenseMatrix` that models a priori densely populated matrices with real coefficients. This class will comply with the following specifications.

Data members :

- `int nr` : the number of rows
- `int nc` : the number of columns
- `std::vector<double> data` a vector storing the entries of the matrix row-wise.

Functions/member functions :

- a constructor `DenseMatrix(const int&, const int&)` intializing `nr,nc` with the input parameters and sizing `data` accordingly, setting the coefficients to 0.
- a copy constructor

- a copy assignment operator `=`
- operator `( , )` that takes a pair of integers (j, k) and returns the (r-valued) coefficient located at the j-th row and k-th column. This operator should be implement both in `const` and non-`const` version.
- output stream operator `<<`
- operator `+` that adds two `DenseMatrix`
- operator `+=` that increments a `DenseMatrix` with another `DenseMatrix`
- operator `*` that multiplies two `DenseMatrix`
- operator `*` that multiplies a `double` (left op.) and a matrix (right op.)
- operator `*` that multiplies a `DenseMatrix` by a `std::vector<double>`
- operator `*=` that right multiplies a `DenseMatrix` by another `DenseMatrix`
- operator `*=` that right multiplies a `DenseMatrix` by a `double`
- friend function `int NbRow(const DenseMatrix&)` that returns the number of rows
- friend function `int NbCol(const DenseMatrix&)` that returns the number of columns

Exercise 3 : map based sparse matrices

Write a class called `MapMatrix` that models sparse matrices with real coefficients. This class will comply with the following specifications.

Data members :

- `int nr` : the number of rows
- `int nc` : the number of columns
- `typedef std::tuple<int,int> NxN` : a type alias
- `std::map<NxN, double> data` : stores the (a priori) non-zero coefficients of the matrix as triples (j, k, v) =(row position, column position, value).

Functions/member functions :

- a constructor `MapMatrix(const int&, const int&)` intializing `nr,nc` with the input parameters.
- a copy constructor
- a copy assignment operator `=`
- a member function `insert(const int& j, const int& k, const double& v)` that adds the value `v` to the coefficient at position `(j,k)` in the matrix
- output stream operator `<<`
- operator `+` that adds two `MapMatrix`
- operator `+=` that increments a `MapMatrix` with another `MapMatrix`
- operator `*` that multiplies two `MapMatrix`
- operator `*` that multiplies a `double` (left op.) and a matrix (right op.)

- operator `*` that multiplies a `MapMatrix` by a `std::vector<double>`
- operator `*=` that right multiplies a `MapMatrix` by another `MapMatrix`
- operator `*=` that right multiplies a `MapMatrix` by a `double`
- friend function `int NbRow(const MapMatrix&)` that returns the number of rows
- friend function `int NbCol(const MapMatrix&)` that returns the number of columns