```cpp
#ifndef METNUM_TP2_LINEARALGEBRA_H
#define METNUM_TP2_LINEARALGEBRA_H

#include <vector>
#include <cmath>
#include <queue>
#include <functional>
#include <list>
#include <numeric>
#include <algorithm>
#include <sstream>
#include "Matrix.h"

#define POWER_ITERATION_DELTA 0.0001
template <typename T>
using min_queue = std::priority_queue<T, std::vector<T>, std::greater<T>>;
// Matriz de datos, vector, número de línea.
typedef std::function<double(const Matrix &, int, const Matrix &, int)> DistanceF;
typedef std::function<double(const std::vector<double> &)> Norm;
typedef std::pair<double, std::vector<double>> EigenPair;

std::vector<double> operator*(const Matrix &m, const std::vector<double> &n);
std::vector<double> operator*(const double &m, const std::vector<double> &n);
EigenPair powerIteration(const Matrix &, std::vector<double>, const Norm &, unsigned int iterations);
void deflation(const Matrix &A, const EigenPair &eigen);
std::list<EigenPair> decompose(Matrix, int, const Norm &, unsigned int iterations);
void dimensionReduction(const Matrix& src, Matrix& dst, const std::list<EigenPair>& l);

const DistanceF L2 = DistanceF([](const Matrix &A, int i0, const Matrix &B, int i1) -> double {
    if (B.columns() != A.columns()) {
        std::stringstream fmt;
        fmt << "Tamaño de matriz A es " << A.columns() << ", mientras que B es " << B.columns();
        throw new std::out_of_range(fmt.str());
    }

    double output = 0.0;

    for (int j = 0; j < A.columns(); ++j) {
        output += std::pow(A(i0, j) - B(i1, j), 2.0);
    }

    return std::sqrt(output);
});

const Norm N2 = Norm([](const std::vector<double> &v) -> double {
    double output = 0.0;

    for (int i = 0; i < v.size(); ++i) {
        output += v[i]*v[i];
    }

    return std::sqrt(output);
});

#endif //METNUM_TP2_LINEARALGEBRA_H
```