Dynamis System Programming HW2 Report

20190837 유지상

In my Matbase class, I used a `std::vector<std::vector<double>>` as the protected member variable to store the matrix/vector values. This is a vector of vectors of doubles. The elements of the outer vector are the rows of the matrix and the elements of the inner vector are the entries of the matrix whose indices represent the columns they belong to.

For large matrices, space complexity becomes an important factor since the memory usage is O(n^2) for a n x n matrix. For a 1000x1000 matrix of doubles, that is 8 megabytes + overhead per matrix. Perhaps, rather than using vectors of vectors or arrays of arrays of fixed sizes, it is better to use sparse matrix representations or memory-efficient storage formats for matrices with many zero elements. Some potential ways of implementing sparse matrix representations include:

- Compressed Sparse Row (CSR) where the matrix is represented using three arrays: values, column indices, and row pointers.
- Compressed Sparse Column (CSC), which is a format similar to CSR, but it stores the matrix in column-major order.
- Coordinate List (COO) where each nonzero element of the matrix is stored as a triple (row index, column index, value).
- Diagonal Storage (DIA) which stores the diagonals of the matrix along with their offsets. It is efficient for diagonal matrices and symmetric matrices.

Performance is also an important factor for performing operations on large matrices. A popular matrix library for computationally efficient programs for scientific applications is Eigen. Eigen uses vectorization and SIMD instructions for fast parallel processing. There is a trade-off, however, between space complexity and time complexity since parallel processing often requires aligned data which is sacrificed when space complexity is prioritized.

Result of simulate: