List of Contract Augmentations

1. Solver classes assume that any matrix passed in is solvable.

2. Changed most of our thrown Runtime errors into ArgumentErrors where appropriate.

3. The Tri-Diagonal Solver class now check the matrix for tri-diagonality.

List of Contract Deviations

1. Removed the abstract classes from our design. Ruby does not have any notion of interfaces, or abstract classes due to its “Duck Typing” feature, and we want our factory and solver classes to be able to support Duck Typing as well.

2. Removed all factory classes from our design. Factory classes made sense in other object oriented languages where they needed to adhere to an interface, so that the compiler knows that the classes are capable of what they set out to do. Ruby does not use interfaces, and as a result, each class’s initialization method is it’s own factory method.

3. Removed all contractual checks that use .kind_of? method. Like change #1, we do not want to limit the use of our library only on types we define.

Additional Notes on Implementation

- For LU Factorization, we used Ruby Matrix’s LUPDecomposition module to solve the problem. The rationale behind this decision was to reuse code already in the native ruby library, and that changing the data structure of the matrix will not have a noticeable effect on the number of floating operations performed (Pivoting + Decomposition + Back Substition still results in $O(n^3)$ time)
- TriDiagonalSolver uses the Thomas algorithm for a solution in time O(n).
- Had we more time, we would have implemented the Conjugate Gradient Method to solve sparse matrices, an iterative algorithm that is designed for sparse matrix linear equation problems.

Additional Testing Beyond Contracts

In order to aid our programming correctness, we have created test cases for all our test classes. We aim to achieve 100% code coverage with our tests. Our tests are set up to use the "SimpleCov" gem, which provides HTML output showing our coverage statistics. Our tests are linked together in the `master_test_suite.rb` file, which includes a reference to all our testing code.

Test Cases:

SparseMatrix Class --

1. Constructor with valid matrix input
2. Constructor with all zeroes matrix input (should return argument error)
3. Constructor with improper type (should return argument error)
4. Adding a new value with valid inputs
5. Adding a new value overwriting an existing input (should return argument error)
6. Adding a new value that is out-of-bounds (should return argument error)
7. Adding a new value that is value 0 (should return argument error)
8. Equality operator implementation
9. Hash operator implementation

LU Solver Class --

1. Correct results for LU Solver on a 3x3 matrix
2. LU Solver on larger input size of b (should return argument error)
3. LU Solver on smaller input size of b (should return argument error)

TriDiagonal Solver Class --

1. Correct results for TriDiagonal Solver on a 4x4 tri diagonal matrix
2. Larger input size of b (should return argument error)
3. Smaller input size of b (should return argument error)