Project 1 Part 1: Design Rationale and System Research

ECE 421 Project 1
Object-Oriented Library for Sparse 2D Matrices
February 2, 2016

Michelle Mabuyo
Kirsten Svidal
Anju Eappen

1. What is a sparse matrix and what features should it possess?

   A sparse matrix is matrix in which most of the elements are zero [1,2]. It allows special techniques to take advantage of the large amount of zero elements [3]. The term "large amount of zeroes" can be interpreted loosely but for our purposes, we define it by the sparsity (fraction of non-zero elements over the total number of elements) being lower than 0.5. Storage, operations and manipulations of sparse matrices can be optimized to take advantage of the amount of zeroes in the matrix.

2. What sources of information should you consult to derive features?

   MatLab already has a library for sparse matrices we can consult to derive our features [4] . We can also consult Ruby's Matrix class as a base class for basic matrix functionality. Sources such as Wolfram and Wikipedia give us a better understanding of what other features could be implemented.

3. Do analogies exist? If so, with what?

   A sparse matrix analogy can be made with a line of balls connected by springs, where only adjacent balls are coupled. This is a loosely coupled system [2].

4. Who is likely to be the user of a sparse matrix package?

   Likely users of this sparse matrix package will be data/modelling programmers, those who need calculations involving network connections, physicists and mathematicians.

5. What features are they likely to demand?

   The users described in Q4 are most likely to demand the following features:
   - Matrix creation, manipulation, reordering algorithms and linear algebra [3]
   - Easy and intuitive matrix entry, multiple ways to create a matrix [7]
   - Basic matrix operations (processing all values in a matrix using a single arithmetic operation or function) [5]
   - Specialized matrix functions such as [6]
   - Basic matrix functionality that Ruby's Matrix class implements [8]
   - Matrix updates and access
   - Matrix metadata
   - Solving partial differential equations
   - Error and input validity checks

- Handling different types of elements appropriately (positive, negative, real, non-real, fractions)

6. What is a tri-diagonal matrix?

   A tri-diagonal matrix is “a square matrix with nonzero elements only on the main diagonal and slots horizontally or vertically adjacent the diagonal (i.e., along the subdiagonal and superdiagonal),” [9] It can be used to for linear solving Ax = b. A tri-diagonal systems of equations can be solved using the tri-diagonal matrix algorithm, where the solution can be obtained in $O(n)$ instead of $O(n^3)$ for regular Gaussian elimination. [10]

7. What is the relationship between a tri-diagonal matrix and a generic sparse matrix?

   A tri-diagonal matrix is a type of generic sparse matrix, in which zero values are not located on the diagonal, provided there are a large number of elements in the matrix.

8. Are tri-diagonal matrices important? And should they impact your design? If so, how?

   Yes, they are important. They impact our design in that we know the layout of tri-diagonal matrix and can therefore make computations more efficient. There will be a TridiagonalMatrix class for optimization due to their unique layout.

9. What is a good data representation for a sparse matrix?

   There are multiple ways to represent a sparse matrix: [2]
   - dictionary of keys with keys being the (row, column) and the value being the value of the element. Entries not in the dictionary are considered 0.
   - list of lists. With one list corresponding to a row, containing the column index and the value.
   - coordinate list. This list stores a tuple of coordinates (row, column, value).

   We are going to represent our sparse matrix with three arrays, as per this video on sparse matrices [11]. The three arrays are called values, columns and row_pointers.

   - values is an array containing all the non-zero data, starting from the upper left position in the matrix and reading from left to right down the matrix.

- columns is an array containing the column numbers. Each element in the columns array corresponds to the location of the same indexed element in the values array. Columns are 0-indexed.
- row_pointers is an array with pointers pointing to an index in the values array that indicate to which non-zero element each row starts with.

10. Assume that you have a customer for your sparse matrix package. The customer states that their primary requirements as: for a N x N matrix with m non-zero entries.
    a. Storage should be ~O(km), where k << N and m is any arbitrary type defined in your design.
    b. Adding the m+1 value into the matrix should have an execution time of ~O(p) where the execution time of all method calls in standard Ruby container classes is considered to have a unit value and p << m ideally p = 1

    In this scenario, what is a good data representation for a sparse matrix?

    An ideal data representation of a sparse matrix would be to store the non-zero values in an array and the corresponding row and column in the same index of the two other arrays. Adding a value would mean adding a value, the row value and the column value to the ends of the corresponding arrays. This means adding an element will occur in constant time.

11. Design Patterns are common tricks which normally enshrine good practice.

    a. Explain the design patterns: Delegate and Abstract Factory
    - Delegate: where an object, instead of performing a task it is called upon to do, delegates that responsibility to another helper class [12,13]. It is used as an alternative to inheritance.
    - Abstract Factory: encapsulates a group of individual but related factories that conform to the same interface. Client object does not build objects directly but specifies which factory it will use and calls methods provided by a common interface. [15,16]

    b. Explain how you would approach implementing these two patterns in Ruby
    - Delegate: There are many ways to implement the Delegate pattern in Ruby. We chose to implement using the method_missing method,[14] which delegates methods that are missing from the class to something that responds to the methods specified by the user.

- Abstract Factory: To implement this in Ruby, you need a module for each factory that conform the to the same interface [17].

c. Are these patterns applicable to this problem? Explain your answer.

- Yes, these patterns are applicable to the problem. Ruby already has a class Matrix that can be delegated to by our SparseMatrix class. For Abstract Factory design pattern, the factories would be a LazyMatrixFactory, which the SparseMatrixFactory and TridiagonalMatrixFactory inherit from. Each factory would then create their own version of a matrix.

12. What implementation approach are you using (reuse class, modify class, inherit from class, compose with class, build new standalone class); justify your selection.

We are using a reuse class implementation approach in the sense that we are reusing the Matrix class to be delegated to by our SparseMatrixFactory class. Much of the functionality that we require often need the full version of the matrix, not just our data of the non-zero values. In this case, it is best to let Matrix take care of the functionality and only rewrite methods that can become more efficient with our current matrix storage implementation.

13. Is iteration a good technique for sparse matrix manipulation? Is "custom" iteration required for this problem?

Since a sparse matrix contains a lot of zero elements, we would want a custom iteration that skips over the zero elements to make matrix manipulations and computations more efficient.

14. What exceptions can occur during the processing of sparse matrices? And how should the system handle them?

- Non-real elements in the matrix, i.e. matrix operations on character matrices may not work. The system will have a precondition in the design contract that asserts sparse matrices must be real. If the precondition is not met, the system must throw an error.
- The SparseMatrix object does not have its sparse characteristics anymore, either due to user initialization error or because of matrix operations that have changed its elements. In situations where the sparsity of a matrix has the possibility of changing, the system will check if the matrix is sparse after the operation is applied to the matrix and alert the user if its sparsity characteristics are not met anymore. It will still

allow the operation and the changes to happen and it will be up to the user to determine what to do with the non-sparse result.

- We will allow types that cannot be coerced to an Integer. However, we will allow the Matrix-delegated methods to throw the appropriate errors when Integer operations cannot be applied to these types. (i.e. char)
- Performing operations on two matrices with incompatible dimensions. System should throw an error.

15. What information does the system require to create a sparse matrix object?

Creating a sparse matrix object is similar to creating a Matrix object. Initialization methods available to the Matrix class [8] (such as ::rows, ::columns, ::diagonal, etc) can be used to create a sparse matrix object. The SparseMatrix class may delegate some initialization methods to the Matrix class however it will also do its own customization. The system will basically require either (a) where the non-zero elements are located (pass in the value, row and column arrays). Here the assumption is that the furthest out non-zero element is the largest index on the matrix. Or, (2) the system needs the full matrix, ideally as an array of arrays, from which it will generate the compressed storage format. These two, and other forms of initialization from Matrix that enable the user to easily make a matrix that can be defined as 'sparse' will be included.

16. Remember you are building for a set of unknown customers – what will they want?

Refer to Question 4 and 5 for a more detailed list of answers.

17. What are the important quality characteristics of a sparse matrix package? Reusability? Efficiency? Efficiency of what?

Efficiency is the most important since we need to be efficient in optimizing storage and calculations due to the large amount of zero elements in the matrix. It should also be easy to use when it comes to initialization, access and matrix operations. The package can also be reusable for future developers who want to extend the design to an n-D matrix.

18. How do we generalize 2-D matrices to n-D matrices, where n > 2 – um, sounds like an extensible design?

Suppose we want to add a 3rd dimension to our matrix. To our current implementation, this would simply mean adding another array and indicating the x, y and z positions (formerly called row and column arrays) of each non-zero value. Adding a dimension would take O(n) time to create and populate the array.

Other Design Decisions

- Some methods are acceptable with the native Ruby Matrix class implementation, whereas others are quicker with how we will store the matrix. Therefore, some methods will be overwritten to take advantage of how we currently optimize.
- It will be assumed that the user of the SparseMatrixFactory will have an idea of what exactly a sparse matrix is and realize that it is ideal to have fewer non-zero elements, rather than many. More non-zero elements means more space and less efficiency. SparseMatrixFactory is optimized for truly sparse matrices.
- TridiagonalMatrixFactory methods will have constraints that prevent the user from adding elements to anything other than to extend all three diagonals. The assumption here is that the user will know what a Tridiagonal Matrix is and realize that this class will work to preserve that definition throughout its operation.
- Exponents must be integers, not floats, matrices/vectors, chars.
- Matrix dimensions should take in integer arguments, not floats.
- For our purposes, tridiagonal matrices also have to be sparse; otherwise, matrix operations would not be optimized if it wasn't sparse.

Resources

[1] Matrix Definition (Wikipedia)
[2] Sparse Matrix Definition (Wikipedia)
[3] Sparse Matrix Definition (Wolfram)
[4] Matlab Sparse Matrix Functions
[5] Matlab Matrix and Array Functions
[6] Matlab Creating and Concatenating Matrices
[7] Matlab Matrix Creation Functions
[8] Ruby Matrix Class Documentation
[9] Tridiagonal Matrix Definition (Wolfram)
[10] Tridiagonal Matrix Algorithm Definition (Wikipedia)
[11] How to Store Sparse Matrices Youtube Video (Udacity)
[12] Delegation Pattern (Wikipedia)
[13] Delegation Pattern (Best Software Engineering Practices)
[14] How To Delegate Methods in Ruby
[15] Abstract Factory Pattern (Wikipedia)
[16] Design Patterns in Ruby: Abstract Factory
[17] Creational Design Patterns in Ruby: Abstract Factory