This assignment has 10 tasks.

The goal of the final project is to experience programming in a group. Proceed in the following way

• discuss the mathematical part of the project in the group first until you fully understand the problem

• divide the problem in subproblems

• discuss how the different parts should be tested

• prepare a max 20 min presentation which should present the mathematical background, the organization of your work, your solutions and maybe alternative attempts, which you decided to reject.

Bring your Laptop or an USB stick for the presentation.

Background

Sparse matrices are matrices with a large amount of zeros. If the matrix has very large dimensions (shape), it might not be storable in the computer’s memory if one stores all its elements - even the zeros. Furthermore many linear algebra tasks (e.g. matrix multiplication) can be significantly faster if the large amount of multiplications with zero is avoided.

Dealing with sparse matrices requires two not independent steps

• Organization how the data is stored in an effective way, such that data access is fast and storage minimized

• Development of algorithms for linear algebra operations using this data organization

Sometimes also conversion algorithms are needed.

Python offers two modules for sparse matrix handling; scipy.sparse and pysparse. In this project we use these packages as benchmarks. A quick reference for different sparse matrix representation is found in http://en.wikipedia.org/wiki/Sparse_matrix.
Tasks

You are asked to design a class SparseMatrix and use the methods CSR and CSC described in the link above. To this end, do at least the following steps.

Task 1

Write a class SparseMatrix with an `__init__` method which uses a numpy.array matrix as input and which generates an internal representation of the matrix in CSR form. Furthermore it should set an attribute `intern_represent` to the value ‘CSR’.

Task 2

Furthermore the `__init__` method should create an attribute `number_of_nonzero` reporting the number of nonzero elements.

Task 3

Give the class a method, which allows to change a particular element in the matrix. It should have the element’s indexes `i`, `j` and its value `a_{ij}` as input. If it changes the numbers of nonzero elements it should change the attribute `number_of_nonzero` accordingly.

Task 4

Give the class a method changing the internal representation from CSR to CSC. Don’t forget to also change the attribute `intern_represent`.

Task 5

Write a method checking if two such matrices are (exactly) equal.

Task 6

Extend your `__init__` method by an extra parameter `Tol` which serves as a threshold for converting a given matrix to a sparse matrix. All elements of the original matrix which have an absolute value less than `Tol` should be considered as zeros. Give `Tol` a default value 0.
Task 7

Write a method for elementwise addition of two objects of type SparseMatrix.

Task 8

Write a method which multiplies a SparseMatrix with a numpy.array of rank one (i.e. a vector).

Task 9

Write tests for all that you did.

Task 10

Check which of the modules scipy.sparse or your class performs better, when
- Inserting a new element in a matrix
- Summing up two matrices
- Multiplying a matrix with a vector.

Include if you like even pysparse in your experiments.

In case of problems or questions don’t hesitate to contact one of the teachers.