This section contains experiments to study matrix completion problem along with the performance evaluation and comparison of some existing algorithms in literature viz. SVT, FPCA and OPTSPACE.

The first subsection illustrates experiments conducted to study matrix completion problem, with the help of a heuristic algorithm proposed by Simon funk, where the intention is to understand the bigger picture and getting a feel of how and when matrix completion works and with what accuracy. Subsequent sections contains an evaluation of more formal algorithms viz SVT, OptSpace and FPCA on real datasets, comparing their performance on different dimensions as follows

1. W
2. W
3. W

Algorithm diagram \_ dia

Experiment setup

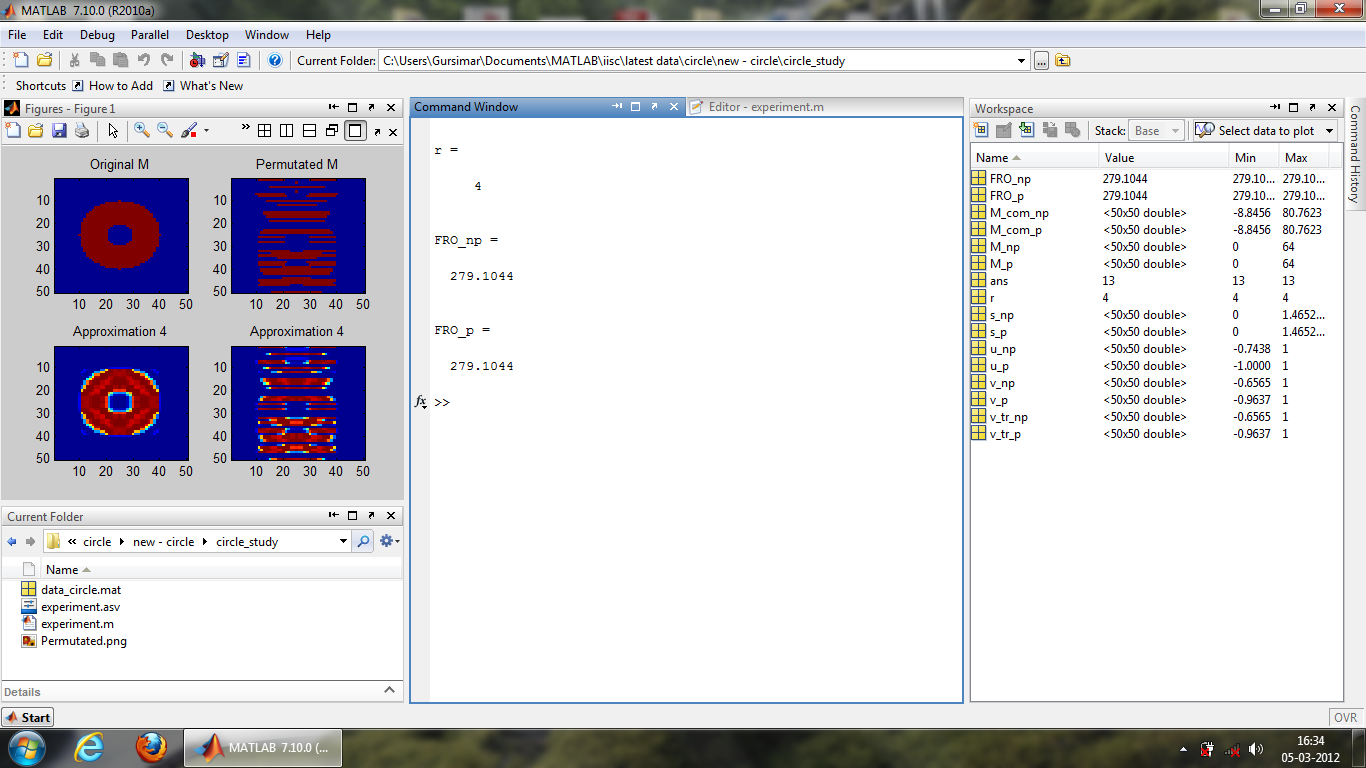


Figure 4.1 - Matlab view of experiments

All the experiments to study verify and compare matrix completion in the following sections is conducted on a machine with the flowing configuration

* Intel i5 processor with 4 real cores and 2 virtual cores through hyperthreading running at 2.53GHz.
* 8GB RAM with 8 GBps memory-processor bandwidth.

Section 1.1

To study how matrix completion works, we need a dataset on which matrix completion algorithm can be applied. Traditional datasets available on internet such as movielens dataset[] contain matrices with large sizes, which are often hard to visualize, leaving us only with the option to rely on a single number metric which depicts approximation accuracy. Although single number metric works perfect for analysis and comparison in majority of technical cases but sometimes it is hard for visualizing the results. The broad objectives of experiments to be performed is to visualize and experiment empirically how and under what conditions matrix completion algorithm will return a decent approximation of the original matrix.

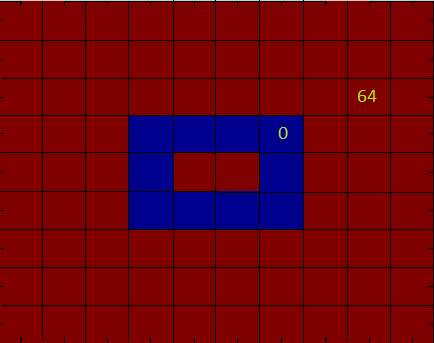
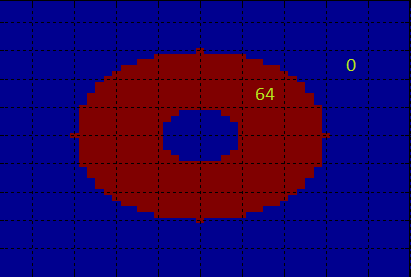
 

Figure 1.1 depicting   
a) 9x10 matrix consisting of binary pattern of 0s and 64s  
b) 50x50 matrix consisting of binary pattern of 0s and 64s.

To serve this purpose we have chosen to work with a specialized pattern of binary low rank matrices containing of 0’s and 64’s as shown in figure 1.1 which are very easy to visualize. In addition for numerically comparing the approximation accuracy, the results in the subsequent sections include empirical curves depicting approximation accuracy.

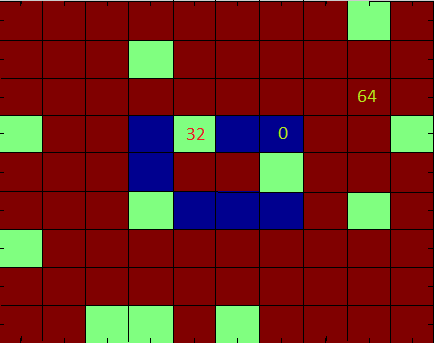
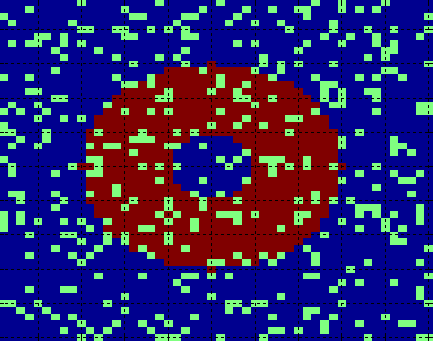
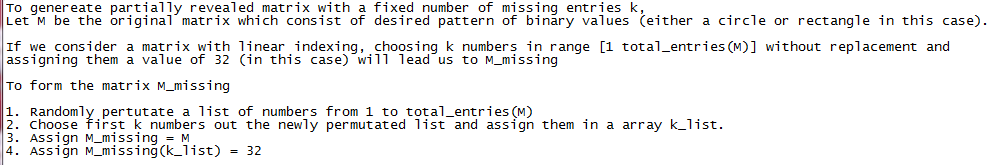
 

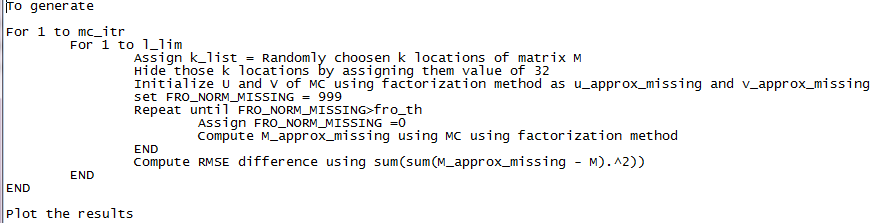
Figure 2.2

Now to generate partially revealed matrices from this original matrix, we randomly pick a fixed number of locations and hide them. As shown in figure 1.2 hidden locations are assigned a value 32, which is entirely for visualisation purposes and is later discarded by the matrix completion algorithm. The algorithm for computing these partially revealed and original matrices is presented here.



As the major objective of the experiments is to get a feel of matrix completion, we have intentionally used an algorithm which is fairly simple to understand. One such algorithm, matrix completion using factorization, presented in section 4.2, satisfies the criteria well and is chosen for these experiments. It must be mentioned here that the algorithm may not be the best choice for solving practical matrix completion problems and many other algorithms exist in literature which may outperform this algorithm in various domains. However our objective here is to understand the nature of matrix completion problem with minimum effort so parameters like convergence time may not be of significant interest here.

We have used matrix completion using factorization algorithm with only a slight modification and a bit different notation. However the way we computed the results, presented in the following sections, needs explicit mention. So the modified algorithm for computing RMSE curves for different values of upper bound of rank is presented in figure 1.3, following which is the snippet of the actual code in figure 1.4. To make things easy to communicate, similar notation as introduced in algorithm is used throughout in the following sections.

We would like to take some time to explain the necessary notation used in the algorithm here along with some data structures and how they fit together. Right in the beginning during initialization we randomly sample two matrices U and V of size mxr and rxn respectively. For a fixed original data matrix M of size mxn, we have the freedom to choose r at our own will which motivates us to think for existence of some optimality in accuracy with r. This motivation leads us to perform experiments whose results are presented in the subsequent sections.

The algorithm gets its first approximation by multiplying U and V (which is u\_approx\_missing and v\_approx\_missing in the code above) to get M’ (m\_approx\_missing). This first approximation of original matrix M is shown in figure 1.8, which moves towards minimizing RMSE difference, improving the approximation with each subsequent iteration.

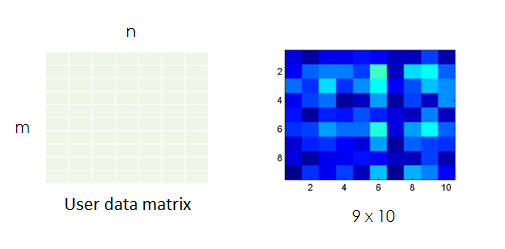


Figure 4.6 - Matrix M' at the start of algorithm

With each iteration the algorithm in section 4.1.2 is expected to converge at a global minimum and we have the best approximation of the original data matrix M. The following illustration, developed in MATLAB illustrates that the optimisation function is convex and we can make it converges to global minimum using some optimisation algorithm such as gradient descent.

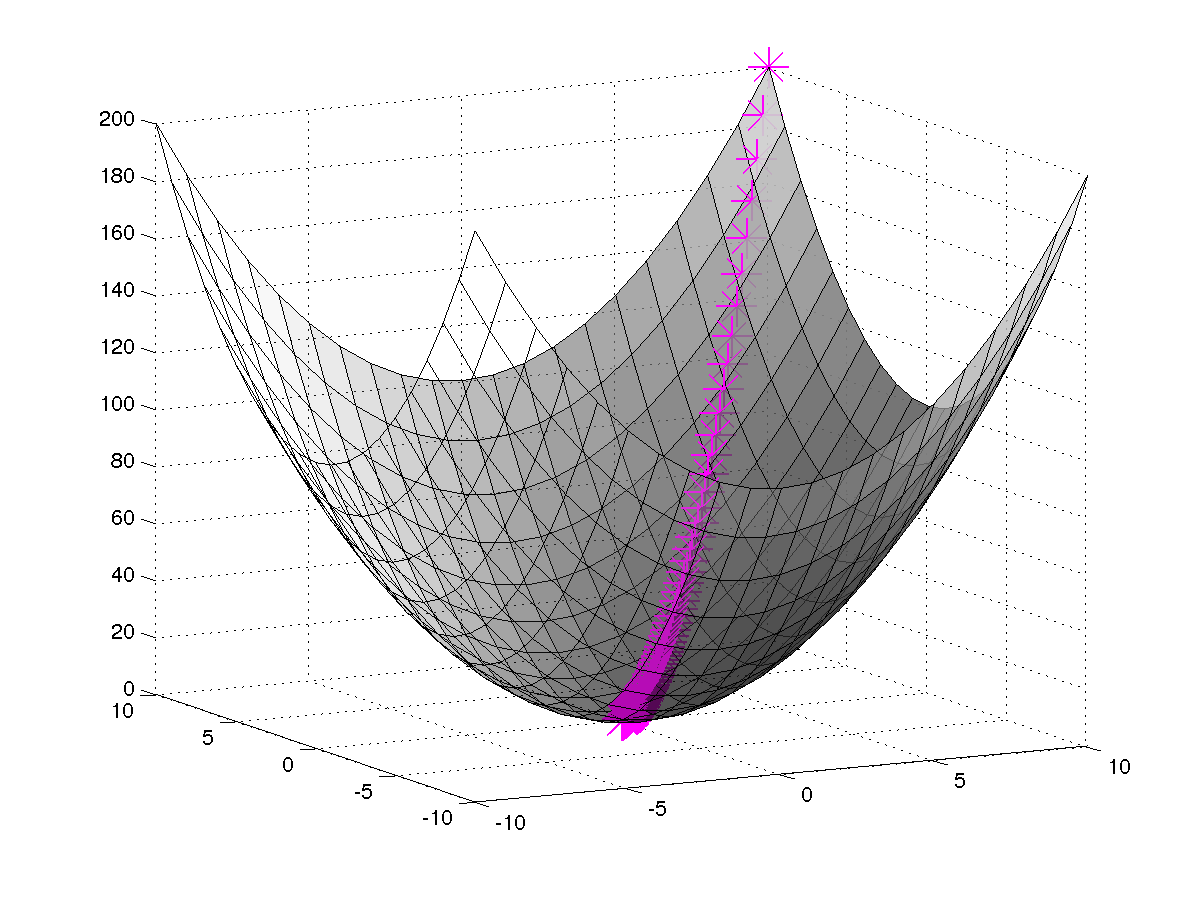


Figure 4.7 - Gradient descent in two dimentions

Things to do – rough work

1. Specify that this section does not focus on exact matrix recovery as candes say…
2. Specify evaluation criteria (RMSE difference)
3. Write proper algorithms
4. Specify notation properly

Results

* single iteration results
* monticarlo iterations
  + fundamental limits
  + permutated results

Single iteration results

The results presented here depict the matrices recovered by the algorithm from some sample test cases. The accuracy of recovery depends on which combination of entries is missing and not just on the number of missing entries. Figure 1.5 illustrates the matrices recovered by the algorithm for same number of missing entries but each case differently located which is chosen at random using the algorithm shown in figure 1.6.

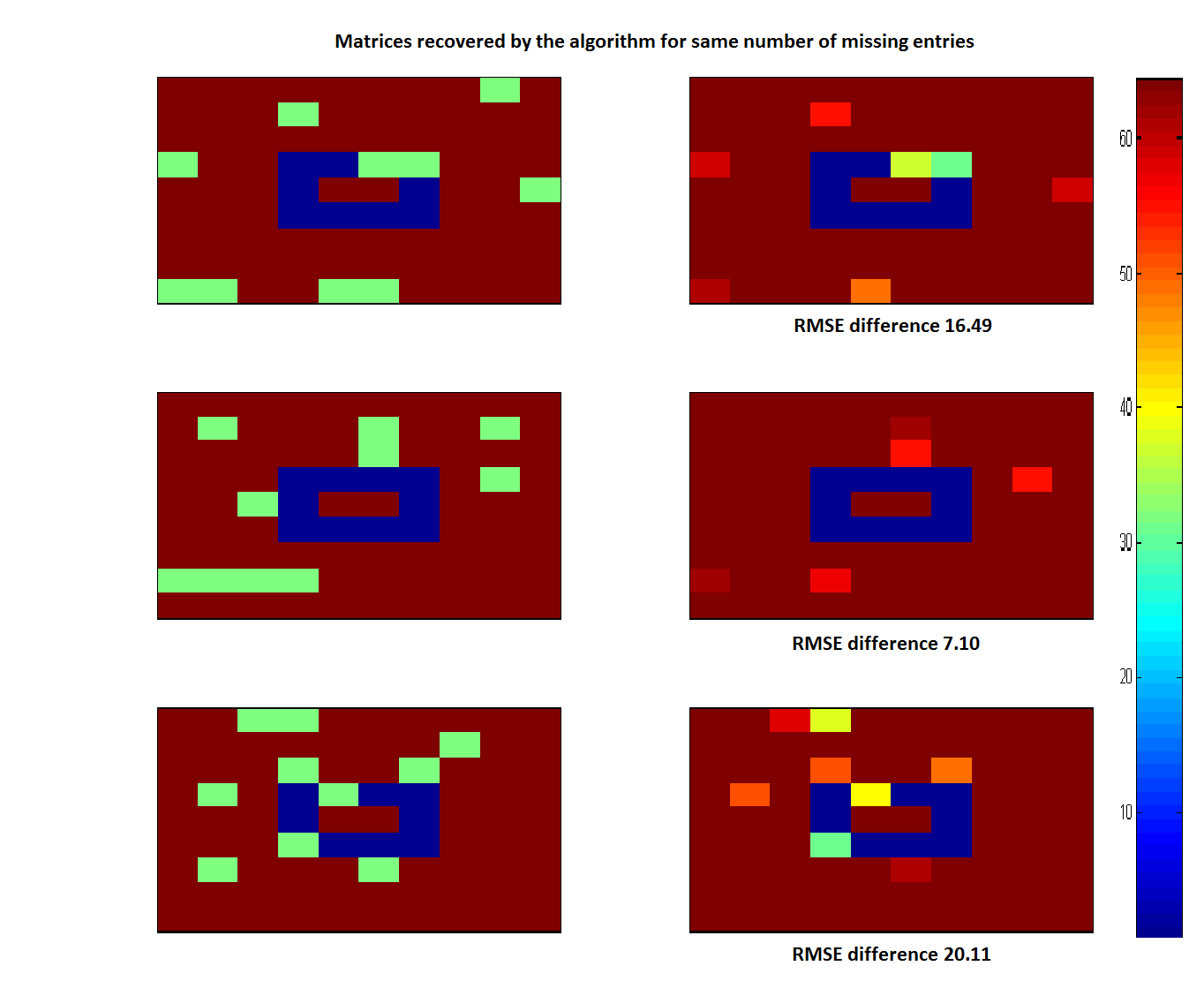
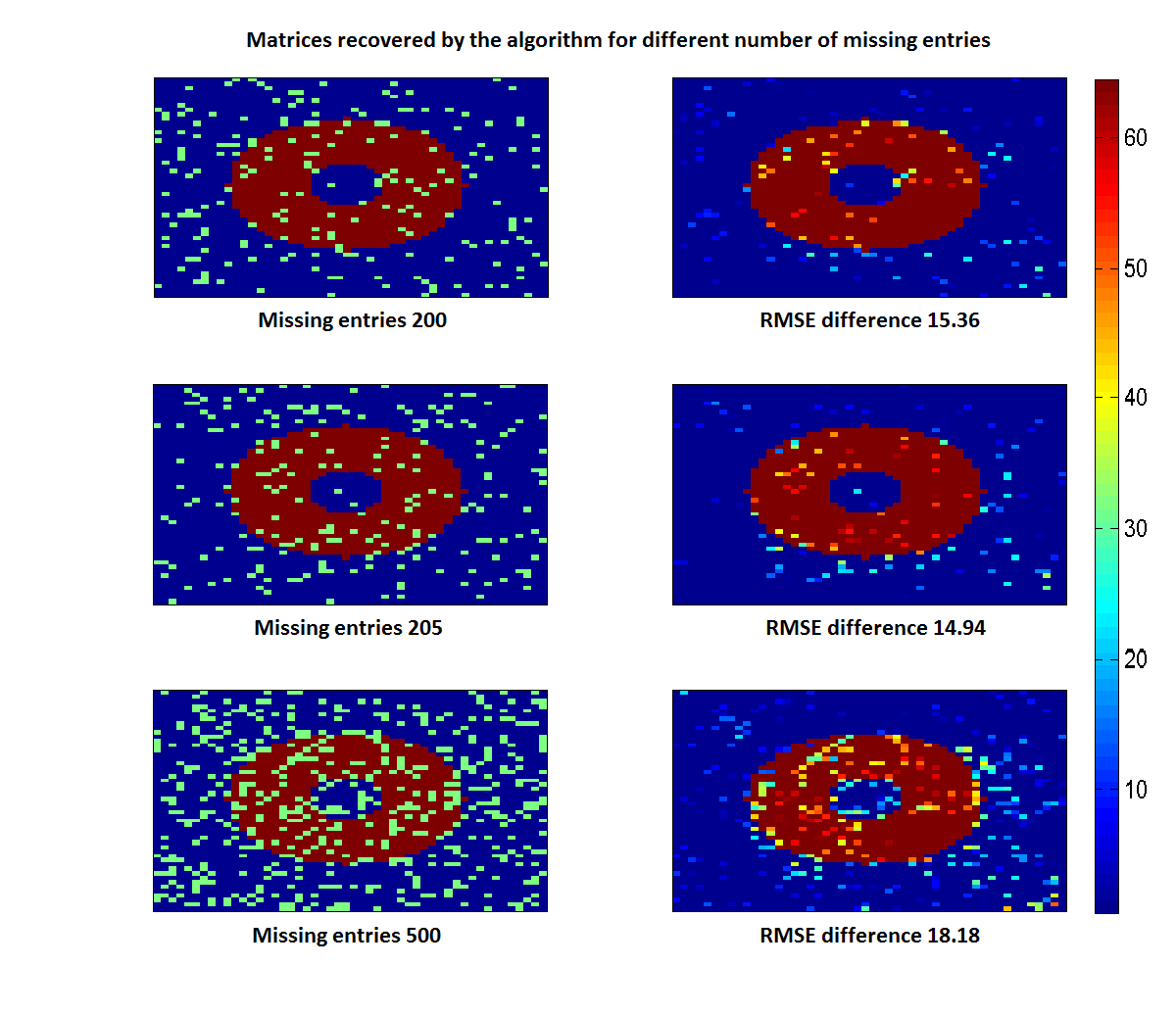


Figure 1.7 Missing entries = 10, iteration= 400

The figure above makes it clear that we can’t judge any algorithm based only on its performance on one single combination of missing entries. Some combinations will perform better than others, as in case of (b) above. Now to verify the effect of number of missing entries, the results presented in figure 1.7, show that the trends are as intended.

The accuracy is expected to decrease as the number of missing entries increase and the trends above give us strong indication it that direction. However the trends do not follow the rule strictly because there is a random component whose performance depends on the combination of missing entries, which is chosen at random. To remove this effect, monti carlo simulations are performed which will average out this random component and we can have a clear view of trends in the results presented in the following sections.

From these results we would immediately like to make some conclusions that will direct the future course of experiments.

1. Matrix completion works only when the original matrix is of low rank or approximately low rank.
2. Accuracy of matrix completion depends significantly on the number of missing entries. More number of missing entries lessens the chances of getting it correct.
3. For fixed number of missing entries, accuracy of matrix completion also depends on which combination of missing entries is used to recover the matrix.

Fundamental limits

The fact that r of U and V is a parameter of the algorithm which we are free to choose, give us some intuition that there may exist some optimal r for which this algorithm may perform best. Thus we choose to perform empirical experiments to find out this fact. However before presenting experimental results, we would like to put emphasis on theoretical fundamental limits on RMSE difference between original matrix and approximated matrix as we vary r. The illustration below

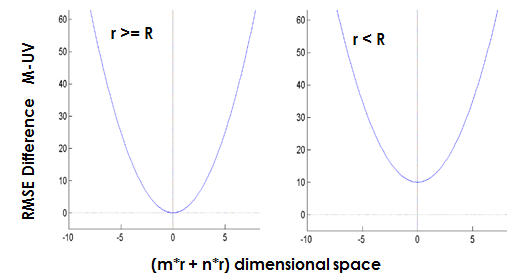


Figure 4.5 - RMSE difference curves with r and R

In the subsequent sections we have conducted experiments for r>R because our interest was to investigate whether some optimal r exist for which matrix recovery is more accurate or efficient. If r<R it is obvious that we can’t recover the matrix exactly, so it is of less practical interest.

It clear from the conclusion #3 that if we want to compare the accuracy of matrix completion by varying various parameters, it may not be a good idea to make conclusion on the basis of merely one combination of missing entries, which is totally random. So it is unwise to make any judgements based on these results, instead we can average results and perform monti-carlo simulations for different combinations of missing entries.

#### Rectangle matrix M

This section contains numerical simulations performed on the matrix M of Figure 4.3. During initialization we have to randomly sample matrices U and V and the corresponding, M’, obtained by multiplying them is displayed below.

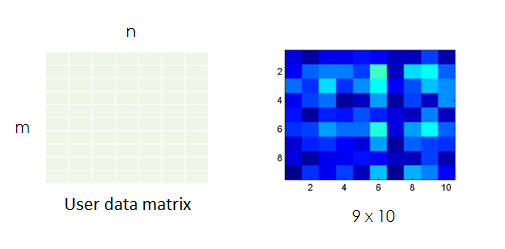


Figure 4.6 - Matrix M' at the start of algorithm

Following which the algorithm in section 4.1.2 is expected to converge at global minimum. The following illustration, developed in MATLAB shows that the optimisation function is convex and how it converges to global minimum.

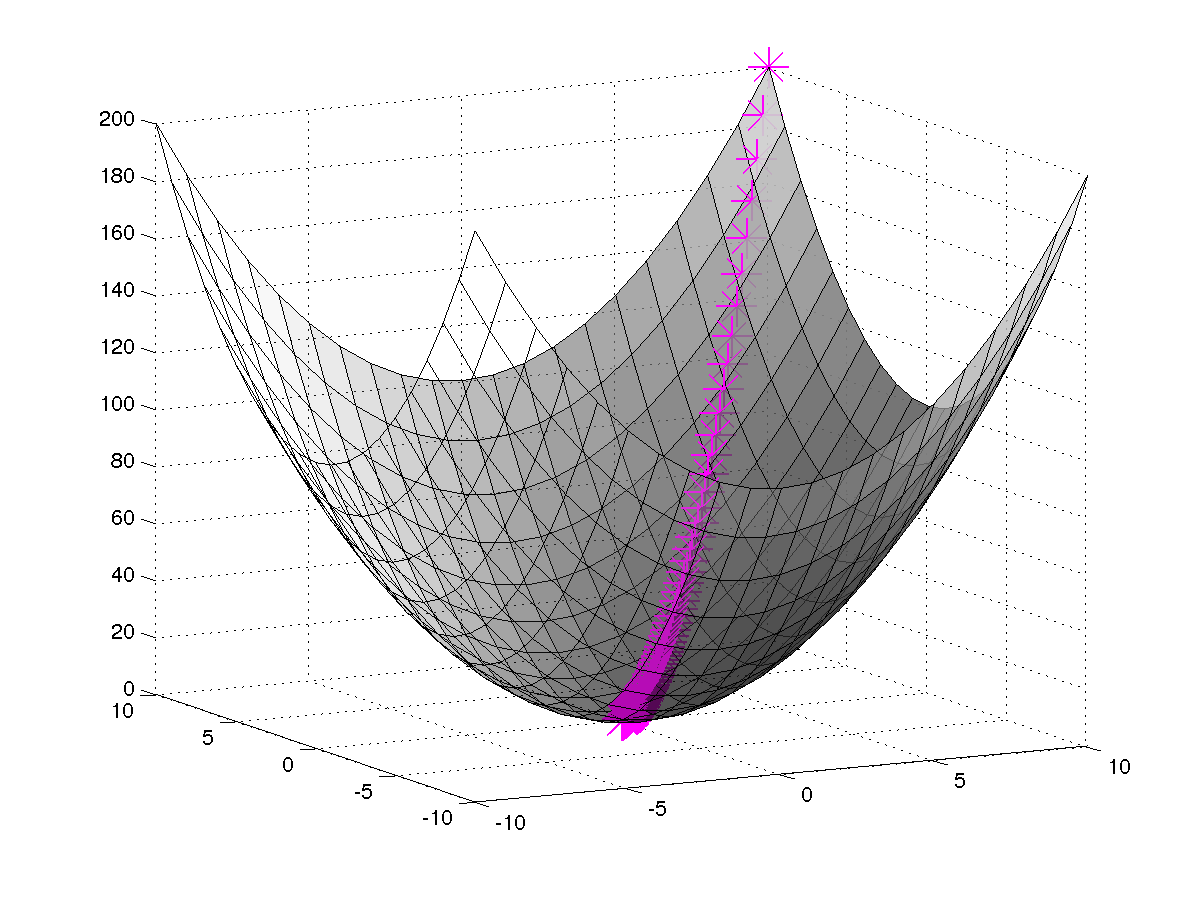


Figure 4.7 - Gradient descent in two dimentions

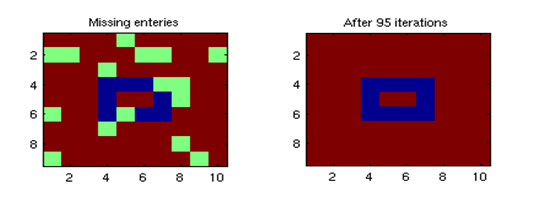


Figure 4.8 - Matrix obtained after completion of algorithm

##### Monti carlo analysis

These curves are obtained by monti-carlo simulations ie by running the same program multiple times and averaging the results. Results comply with theory, if percentage sparseness increases, the error RMSE difference between the two matrices increases. Thus matrix recovery becomes harder.

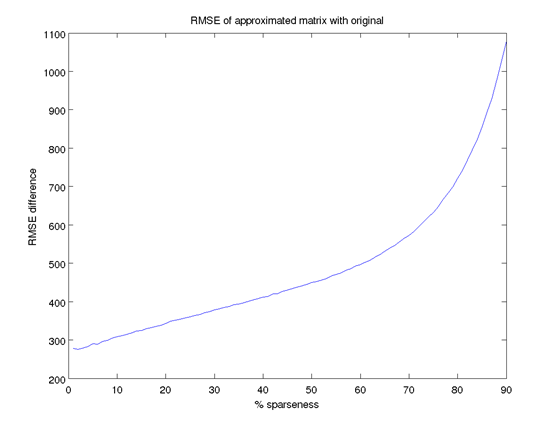


Figure 4.9 - RMSE curve with % sparseness for rectangular matrix

An experiment conducted with **permuted matrix** shows that, even if we permute the matrix we get the same RMSE curve.

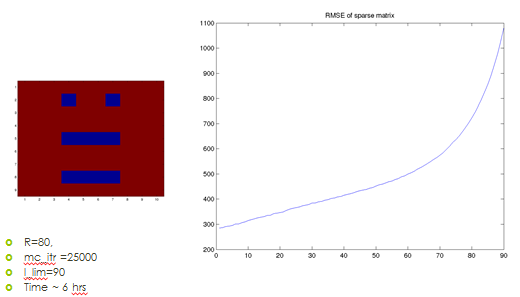


Figure 4.10 - RMSE curve of rectangle permuted matrix



Figure 4.11 - Closer look on RMSE curves

##### RMSE curves with r>R

RMSE curves taken with different upper bound on rank of U and V matrix r (>R) indicates that lower rank facilitates better recovery with this algorithm. However we know due to theory that if r<R error will increase it again as there is a fundamental barrier to matrix recovery. So there should be some optimal rank for which matrix recovery is most accurate, however we observer in subsequent section that we have to spent more computational power to get that result. So time factor increases.



Figure 4.12 - RMSE curves r>R

#### Circle

**Generation of circle matrix**



To save time we obtain curves with an interval of 50, as matrix recovery becomes hard when we have middle rank matrices with large size. Another thing worth noting is that we have same pattern of curves as in case of rectangular matrix.

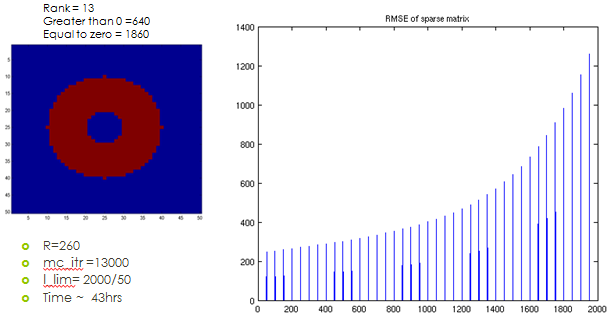


Figure 4.13 - RMSE curve of circle

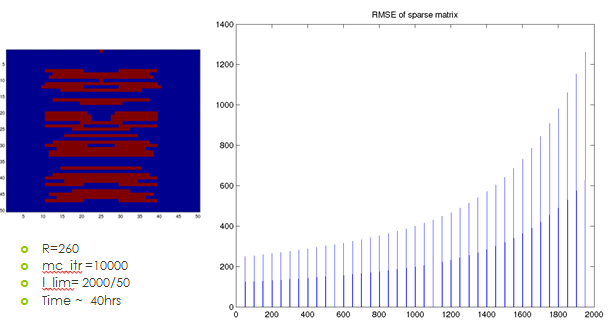


Figure 4.14 - RMSE curve of permuted circle

### Testing

#### Random number generator

This figure indicates a result for simulation obtained by taking a large number of sampling and then averaging before plotting them in a histogram (line here). A approximately horizontal line indicates that distribution from which we are sampling is almost uniform.

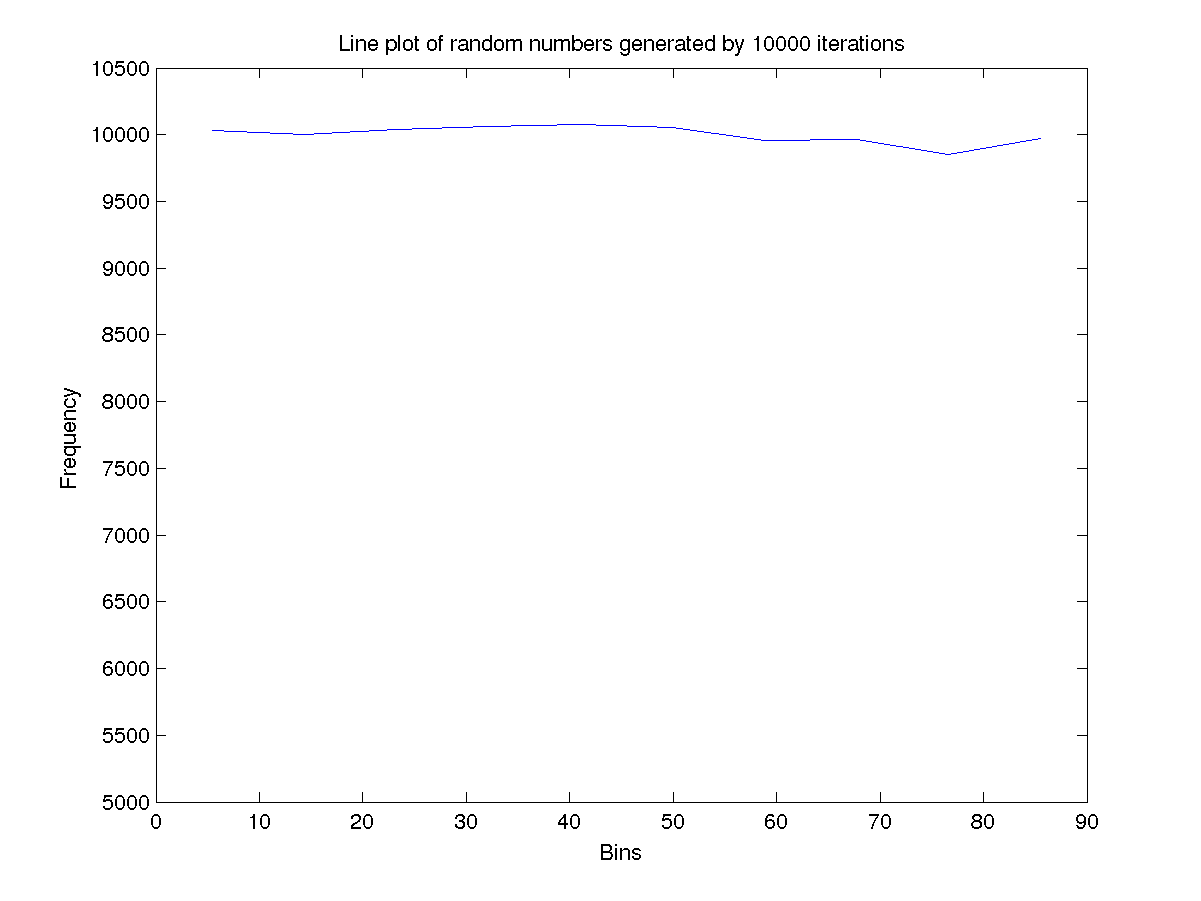


Figure 4.15 - Random number generator distribution testing

#### Convergence time of the algorithm vs rank.

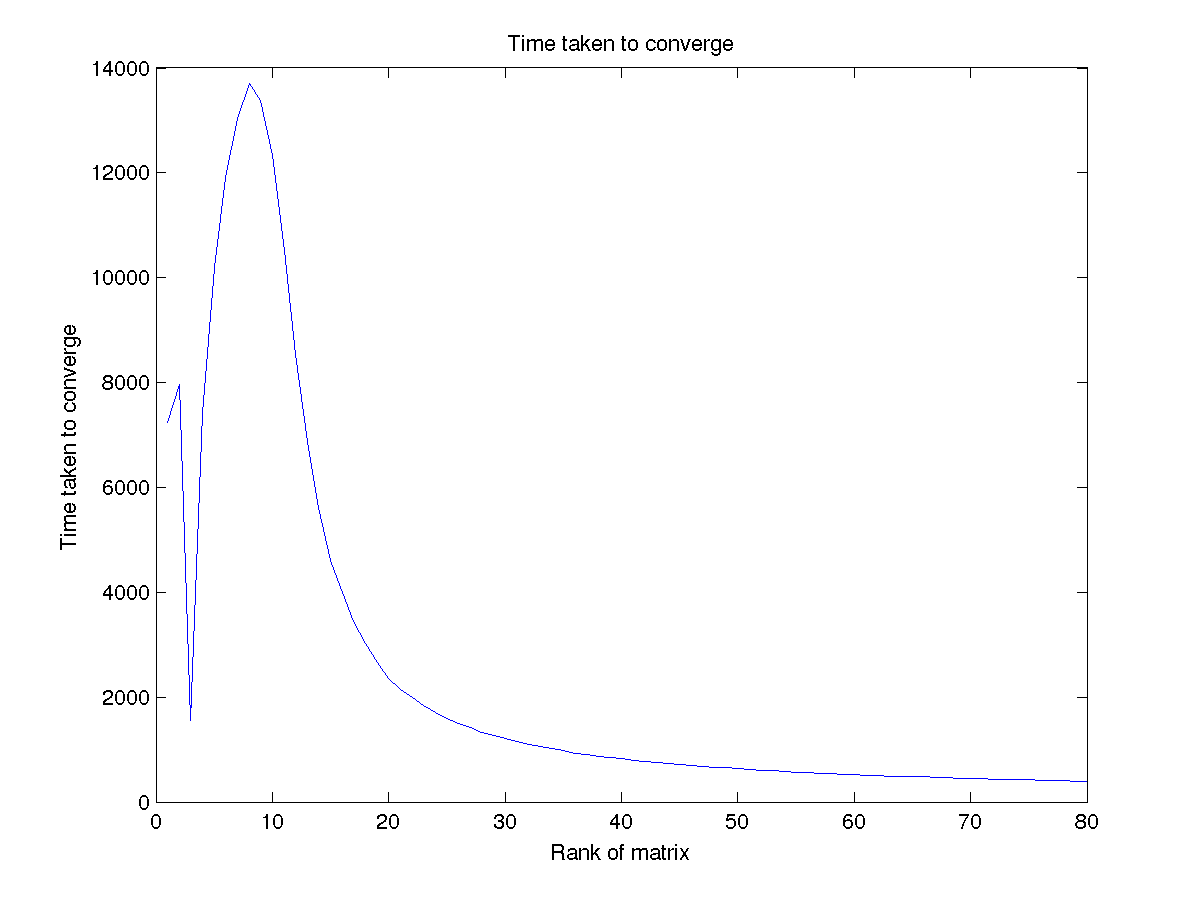


Figure 4.16 - Time vs r curve for RMSE

#### Working of single iteration

Single iteration testing while debugging indicates expected pattern. For a matrix with no missing entries we have RMSE of almost 0 while for the matrix with 20% sparseness we have spikes which can be explained with the fact that different combinations of missing entries may result in different accuracy of matrix recovery as explained earlier.

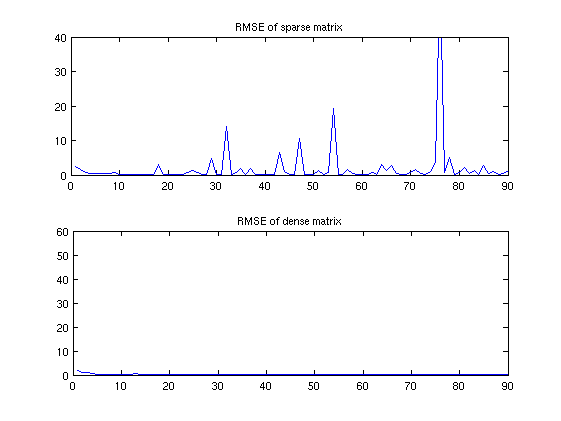


Figure 4.17 - Single iteration testing

Section 2

A very simple and novel matrix with patterns of 0s and 64s is chosen that can be visualized easily using the *imagesc* function of the matlab. However it tunes out that many of algorithms like FPCA and SVT deisgned for specialized problems doesn’t work well with such patterns so in order for comparison we have used the real datasets.

Conclusion

1. Add that rank of matrix should be less than max(m,n) for matrix completion to work.