Karlsruher Institut für Technologie

Final presentation

Automated test matrix generation

Anna Katharina Ricker

Supervisors
Hartwig Anzt Markus Götz

August 24, 2019

Contents

1 Introduction

For the use case of generating matrices for training a neural network to classify, which solver is the best, it is not mandatory needed to generate a matrix with every tiny implementation option (e.g. generate Triangular upper or lower bandwidth) detailed chosen. It is rather usefull to have many different matrices close to real world matrices. Because it was hard to find specific types of real world matrices, this generator is strongly oriented on LAPACK [1]. LAPACK offers a random matrix generator, that was developed to generate matrices to test large numerical linear algebra libraries.

2 LAPACK

LAPACK deals with the implementation of a suite of test matrix generators for testing linear algebra software. Their routines generate random matrices with certain properties which are useful for testing linear equation solving, least squares, and eigendecomposition software. These properties include the spectrum, symmetry, bandwidth, norm, sparsity, conditioning (with respect to inversion or for the eigenproblem), type (real or complex), and storage scheme (dense, packed, or banded). The three main routines they have developed are called xLATMR, xLATMS and xLATME, where the first letter of each name (the 'x') is either 'S', 'D', 'C', or 'Z'.

- 'S' for single precision real (REAL)
- 'D' for double precision real (DOUBLE PRECISION)
- 'C' for single precision complex (COMPLEX)
- 'Z' for double precision complex (DOUBLE COMPLEX)

Since complex matrices would go beyond the scope of our work, I will focus myself on the LAPACK -implementation of real matrices:

2.1 Distribution option in LAPACK

They use three different distributions:

- a uniform distribution on (0; 1)
- a uniform distribution on (1; 1)
- a normal distribution with mean zero and variance one (normal(0; 1))

2.2 Diagonalentries options in LAPACK

To generate Diagonal entries, LAPACK always uses one of the following options:

- Input by the user.
- D(1) = 1 and the other D(i)=1/COND, where COND >= 1 is a user input.
- D(N)=1/COND and the other D(i) = 1.

- The D(i) form a geometric sequence from 1 to 1/COND.
- The D(i) form an arithmetic sequence from 1 to 1/COND
- The D(i) are random in the range [1/COND; 1] with uniformly distributed logarithms.
- The D(i) are random with the same distribution as the other matrix entries.

In addition, each D(i) may optionally be multiplied by a random number with absolute value 1.

2.3 Packing options in LAPACK

The allowable options are:

- no packing,
- zeroing out the upper or lower half (if symmetric or Hermitian),
- storing the upper or lower half columnwise (if symmetric, Hermitian or triangular),
- using triangular band storage (upper half or lower half, and only if the matrix is symmetric, Hermitian or triangular), and full band storage.

2.4 the different generators in LAPACK

2.4.1 xLATMR

xLATMR generates a matrix with random off-diagonal entries and given diagonal entries. It is the simplest (and fastest) of the three routines in the suite, and permits no direct control over the eigenvalues or singular values of the generated matrix.

high level description:

1. Generate a matrix A with random entries with a distribution as described before in distribution-options
2. Set Diagonal of A to D where entries D are computed as explained before
3. Grade A, if desired, by pre- and postmultiplying it by diagonal matrices DL and DR, respectively. The entries of DL and DR are chosen just as the entries of D above
4. Permute the rows and columns of A, if desired.
5. Set random entries of A to zero, if desired, to get a matrix with a given fraction of zero entries.
6. Make A a band matrix, if desired, by zeroing out its entries outside given upper and lower bandwidths.
7. Scale A, if desired, to have a given maximum absolute entry.
8. Pack A, if desired

2.4.2 xLATMS

xLATMS generates either a random real symmetric or complex Hermitian matrix with given eigenvalues and bandwidth, or a random nonsymmetric or complex symmetric matrix with given singular values and upper and lower bandwidth.

high level description:

1. Set the diagonal of the matrix A to D, where the entries of D can be chosen as explained in chapter "Diagonalentrie options". The entries of D will be the eigenvalues (and/or singular values) of the final matrix.
2. Pre- and postmultiply A by random orthogonal matrices (if A is real). If A is to be symmetric, then the premultiplying matrix is the transpose of the postmultiplying one. Otherwise, the pre- and postmultipying matrices are chosen independently of one another.
3. Reduce A to have the desired bandwidth using Householder transformations.
4. Pack A, if desired with the described packing-options

For non-banded and large-bandwidth matrices, the preceding description accurately describes the procedure actually followed. For small-bandwidth matrices, steps 2-4 are replaced by a method which uses Givens rotations to increase the bandwidth to the desired value, rather than generating a dense matrix and then reducing its bandwidth. Givens rotation method is used, if:

- the matrix is symmetric and the bandwidth (KL or KU) is less than N/2,
- the matrix is non-symmetric and KL + KU $<$ 0,3(M + N)

With KL being the lower bandwidth and KU the upper bandwidth.

2.4.3 xLATME

xLATME generates a random nonsymmetric matrix with given eigenvalues, either a given condition number for the eigenvalues or a given Jordan form (with certain restrictions), and a given one-sided bandwidth. Thus, for example, they can generate random Hessenberg matrices with given eigenvalues and sensitivities; this is useful for testing QR iteration algorithms for the nonsymmetric eigenproblem. Only dense storage is provided, since the nonsymmetric eigenroutines only work on matrices in dense storage format.

high level description:

1. Set the diagonal of the matrix A to D, where the entries of D can be chosen as explained in chapter "Diagonal-entry options".
2. If the user so specifies, the upper triangle of A is filled with random numbers, which are chosen with a distribution described before in distribution-options. This option may be used to partially control the Jordan form of A as follows: if A has any multiple eigenvalues and the upper triangle is filled in, then there will be exactly one Jordan block per distinct eigenvalue; such a matrix is called defective. If the upper triangle is not filled in, there will only be 1 by 1 blocks in the Jordan form, even if there are multiple eigenvalues; such a matrix is called derogatory.

3. If the user so specifies, A is premultiplied by a random matrix S and postmultiplied by S^{-1}. Here S is a random dense nonsymmetric matrix whose singular values DS may be chosen as described in "Diagonal-entry-options". This option may be used to control the condition of A's eigenproblem as follows: If the upper triangle has not been filled in, then the most sensitive eigenvalue of A will have sensitivity approximately equal to $\kappa \equiv max_i|DS(i)|/min_i|DS(i)|$

   The approximation arises because the true sensitivity need only be within a factor N (the dimension of A) of the condition number of SX, where X is a diagonal matrix chosen so that the columns of SX have equal norm. In general, the columns of S will not differ too much in norm, so that the condition number of the eigenproblem may indeed be approximately controlled with this approach.

4. If the user so specifies, either the upper or lower bandwidth (but not both) is reduced to any positive value desired.

5. Scale A, if desired, to have a given maximum absolute entry

3 Automated test matrix generation

The matrix generator should be easy to extend. That worked with defining different generator options similar as the LAPACK approach, which can be choosen by the user or random. Therefore I implemented 6 matrix-generators and 6 diagonal-entry generators.

3.1 diagonal generator

The approaches are adopted from LAPACK. But it is not possible to set diagonal entries by user-input. That is why this generator has just 6 different diagonal-entry setters:

1. D(1) = 1 and the other D(i)=1/COND, where COND >= 1 is a user input.

2. D(N)=1/COND and the other D(i) = 1.

3. The D(i) form a geometric sequence from 1 to 1/COND.

4. The D(i) form an arithmetic sequence from 1 to 1/COND

5. The D(i) are random in the range [1/COND; 1] with uniformly distributed logarithms.

6. The D(i) are random with the same distribution as the other matrix entries.

if not the user input is defined by zero, a diagonal generator is randomly chosen

3.2 matrix-generators

3.2.1 Generator 1

This generator is based on xLATMS:

1. Set the diagonal of the matrix A to D, where the entries of D can be chosen as explained in chapter "Diagonal-entry options". The entries of D will be the eigenvalues (and/or singular values) of the final matrix.
2. Pre- and postmultiply A by random orthogonal matrices (if A is real). If A is to be symmetric, then the premultiplying matrix is the transpose of the postmultiplying one. Otherwise, the pre- and postmultiplying matrices are chosen independently of one another.
3. Reduce A to have the desired bandwidth using Householder transformations [2].

As it is usefull to save all matrices from different generators and different types in one hdf5 file, there will be no packing options available in this approach. To have an orthogonal matrix I can multiply matrix A with, I created Generator 6, that just creates orthogonal matrices. I thought it could be a good idea to have the possibility to just create a pure orthogonal matrix by its own.

3.2.2 Generator 2

This generator is based on xLATME

1. Set the diagonal of the matrix A to D, where the entries of D can be chosen as explained in chapter "Diagonal-entry options".
2. fill randomly rather the upper, the lower or no triangle of matrix A
3. randomly choose, if matrix A will be multiplied by S and S^{-1} or not. S will be generated by generating random matrix entries with distributions as defined in Lapac. After that the diagonal-entries will be set by the diagonal-generator. This step is not part of xLatme, but it is similarity invariant, so by multiplying it keeps some important characteristics of matrix A
4. randomly choose if the upper or lower bandwidth is reduced. The positive value is a random number between 1 and the size of the matrix, with value = 1 meaning, A will be a triangular matrix.
5. randomly create the given density with the density setter. This step is also not part of xLATME but it causes a more slightly occupied matrix.

unlike to xLatme there is no possibility to scale the matrix. all matrix entries are alway between -1 to 1 or 0 to 1.

3.2.3 Generator 3

This generator is not similar to any LAPACK generators. It just sets the diagonal entries with the diagonal-entry generator. Following there will be 0 to 10 side diagonals with a randomly chosen distance filled with values unequal zero. The distribution options for the random values are the same as given by LAPACK.

3.2.4 Generator 4

This generator is based on xLATMR:

1. spread random matrix entries all over the matrix with on of the distribution described before.
2. randomly chose if a diagonal matrix will or will be not pre- or postmultiplied. The diagonal matrix is generated by the diagonal-entry-generator.
3. randomly chose if rows or columns will or will be not permuted.
4. randomly create the given density with the density setter.
5. randomly choose if the bandwidth is reduced. The positive value is a random number between 1 and the size of the matrix, with value = 1 meaning, A will be a diagonal matrix.
6. set diagonal entries at the end, so it is guaranteed, that the diagonal entries do not equal zero.

3.2.5 Generator 5

This generator does just generate orthogonal matrices.

3.2.6 Generator 6

This generator guarantees positive definite matrices. By filling the diagonal entries with only positive entries and premultiplying with a matrix S and postmultiplying by S^{-1}, the multiplication result will be similar to the generated diagonal matrix which is positive definite. The multiplication by a matrix S and S^{-1} will be repeated 1 to 10 times randomly.

3.3 Density setter

First idea:

- x = density * number of matrix entries
- Set random entries of A to zero (x times)

Second idea: Set random entries of A to zero with a given probability (e.g. density = 0,8 means probability of 80

3.4 Side-diagonal matrix

The generator also deals with the topic of side-diagonal matrices based on the problem of connectivity graphs, where it is

3.5 Householder function

3.6 saving

3.7 check components

3.8 How to guaranty regularity in matrix generation?

generator 2 does not guarantee to generate a regular matrix. so the unregular matrices will be sorted out

3.9 User Input

1. matrix size: int
2. density of the matrix: double [0, 1]
3. positive definite: Boolean
4. Distribution: 1-3
5. amount of matrices: int
6. storage location: String
7. diagonal option: int between 1 and 6
8. generation option: int between 1 and 6

4 References

References

[1] James Demmel Alan McKenney. *LAPACK Working Note 9 ATest Matrix Generation Suite.*

[2] Marlis Hochbruck *Numerische Mathematik I und II WS 2018/19 – SS 2019.* (German)

[3] Answere 1
`https://stackoverflow.com/questions/38426349/how-to-create-random-orthonormal-matrix-in`