Università della Svizzera italiana

Efficient Computational Algorithms 2021

Final Project Report

Decomposition Algorithms

Abstract

This project outlines the factorization of matrices with the six most widely used decomposition algorithms according to [1]

Student 1
Muhammet USLU

Student 2
Mao (Yen) PO

1 Introduction

We use the terms decomposition and factorization interchangeably to mean writing a matrix as a product of two or more other simpler matrices, generally with some defined properties (such as lower/upper triangular).
The underlying principle of the decomposition approach to matrix computation is that to construct computational platforms from which a variety of problems can be solved. The article deals primarily with dense matrix computations. Although the decomposition approach has greatly influenced iterative and direct methods for sparse matrices.

Benefits of the decomposition approach :

- Matrix decomposition solves not one but many problems.
- Matrix decomposition, which is generally expensive to compute, can be reused to solve new problems involving the original matrix.
- The decomposition approach facilitates rounding-error analysis.

We will focus on six most widely used algorithms namely; Cholesky, LU, Spectral, SVD, Schur, QR. For sure these are not the only used methods.

2 Cholesky

If a matrix M is symmetric ($\mathbf{M^T} = \mathrm{M}$), positive semi-definite (all eigenvalues are non-negative) or a complex Hermitian matrix ($\mathrm{M} = \overline{M}^{\mathbf{T}}$) then we can use $\mathbf{LL^T}$ factorization such that L is a lower triangular matrix. [2]

$$\mathrm{M} = \mathrm{L} * \mathbf{L^T} \longrightarrow \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{bmatrix} = \begin{bmatrix} l_{1,1} & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} \end{bmatrix} * \begin{bmatrix} l_{1,1} & l_{1,2} & l_{1,3} \\ 0 & l_{2,2} & l_{2,3} \\ 0 & 0 & l_{3,3} \end{bmatrix}$$

$$= \begin{bmatrix} l_{1,1}^2 & l_{1,1} * l_{2,1} & l_{1,1} * l_{3,1} \\ l_{2,1} * l_{1,1} & l_{2,1}^2 + l_{2,2}^2 & l_{2,1} * l_{3,1} + l_{2,2} * l_{3,2} \\ l_{3,1} * l_{1,1} & l_{3,1} * l_{2,1} + l_{3,2} * l_{2,2} & l_{3,1}^2 + l_{3,2}^2 + l_{3,3}^2 \end{bmatrix}$$

So we can generalize the formulas for higher dimensional matrices...

$l_{i,i} = \sqrt{m_{i,i} - \sum_{j=1}^{i-1} l_{i,j}^2}$ for i = 1,2,... n and for i > j $l_{i,j} = \frac{m_{i,j} - \sum_{k=1}^{j-1} l_{i,k} * l_{j,k}}{l_{j,j}}$ with i = 1,2,... n and j = 1,2,... i-1

The Computational Cost: Computing diagonal element $l_{i,i}$ requires

- i subtractions
- i - 1 multiplications, and
- one square root,

which adds up to 2i operations so overall diagonal element calculation costs us $\sum_{i=1}^{n} 2i = 2\,\frac{n*(n+1)}{2} = n^2 + n$

On the other hand $l_{i,j}$ requires

- j subtractions
- j - 1 multiplications, and
- one division,

so totally 2j operations, then for n dimensional matrix it will require $\sum_{i=1}^{n} \sum_{j=1}^{i-1} 2j = \sum_{i=1}^{n} 2\frac{(i-1)*i}{2} = \frac{n^3}{3} - \frac{n}{3}$ floating-point operations.

Finally summing up those two costs results by the domination of $\frac{n^3}{3}$ Flops

Note : if M is very sparse, most of elements of the matrix are zero , L is often (but not always) sparse then if L is

sparse, the cost of the factorization is much less than $\frac{n^3}{3}$
Cholesky factorization is numerically stable; instability is characterized by significant changes in the predicted dynamical behaviour induced by only small changes in the specification of the system.

Application: Quantum Chemistry; cholesky decomposition of the two-electron integral matrix [3]

3 LU

An invertible matrix M has an LU decomposition , L is an lower-triangular matrix with all diagonal entries equal to 1 and U is an upper-triangular matrix, provided that all its leading sub-matrices are non-singular (with non-zero determinants).
The k^{th} leading sub-matrix of M is denoted M_k and is the k×k matrix found by looking only at the top k rows and leftmost k columns.
Thanks to Pivoting (so called row interchange); it is always possible to re-order the rows of an invertible matrix so that all of the sub-matrices have non-zero determinants. That is why this decomposition is also called LUP factorization, P is the permutation matrix. [4]
Let us assume there is no pivoting so the P = I

$$\text{M = L * U * P} \longrightarrow \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{2,1} & 1 & 0 \\ l_{3,1} & l_{3,2} & 1 \end{bmatrix} * \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ 0 & 0 & u_{3,3} \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ l_{2,1} * u_{1,1} & l_{2,1} * u_{1,2} + u_{2,2} & l_{2,1} * u_{1,3} + u_{2,3} \\ l_{3,1} * u_{1,1} & l_{3,1} * u_{1,2} + l_{3,2} * u_{2,2} & l_{3,1} * u_{1,3} + l_{3,2} * u_{2,3} + u_{3,3} \end{bmatrix}$$

Generalized formulas :
$u_{ij} = m_{ij} - \sum_{k=1}^{i-1} u_{kj} * l_{ik} \quad for \quad j = 1, 2, ...n \quad i = 1, 2, ...j+1 \quad k = 1, 2, ...i$
$l_{ij} = \frac{1}{u_{jj}} * (m_{ij} - \sum_{k=1}^{j-1} u_{kj} * l_{ik} \quad and \quad l_{j,j} = 1 \quad for \quad j = 1, 2, ...n \quad i = j, j+1, ...n \quad k = 1, 2, ...j$
The total cost is $\frac{2}{3}n^3$ Flop

Some features of LU Decomposition:

• For some choices of P, small rounding errors in the algorithm cause very large errors in the solution (numerical instability).
• when P permutes the largest element (absolute value) of first column of M to the 1,1-position.

Application: Power Engineering; Power Flow Computation [5]

4 SPECTRAL

The Spectral Decomposition recasts a matrix in terms of its eigenvalues and eigenvectors. This representation turns out to be enormously useful.
Each symmetric M (nxn) matrix can be written as PDP^{-1} or PDP^T, where D is $diag(\lambda_1, ..., \lambda_n)$, λ_i is the corresponding eigenvalue and where P (Jordan canonical form)is an orthogonal matrix, $(v_1, ..., v_n)$, consisting of the orthonormal eigenvectors v_i. One important side note is orthogonal matrices have unit norm. [6]

$$\text{M} = \text{P} * \text{D} * P^T \quad \longrightarrow \quad \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{bmatrix} = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ v_1 & v_2 & v_3 \\ \downarrow & \downarrow & \downarrow \end{bmatrix} * \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} * \begin{bmatrix} \leftarrow & v_1 & \rightarrow \\ \leftarrow & v_2 & \rightarrow \\ \leftarrow & v_3 & \rightarrow \end{bmatrix}$$

Generalized formula :

$\text{M} = \sum_{i=1}^{n} \lambda_i * v_i * v_i^T \qquad for \quad i = 1, 2, ...n$

Some Usefull features of Spectral Factorization:
∗ The Jordan decomposition gives a representation of a symmetric matrix in terms of eigenvalues and eigenvectors.
∗ The eigenvectors belonging to the largest eigenvalues indicate the “main direction” of the data.
∗ The Jordan decomposition allows one to easily compute the power of a symmetric matrix $M^\alpha = P * D^\alpha * P^T$

Note : The Spectral Decomposition is a special kind of Eigenvalue Decomposition in which the matrix P is orthogonal which means that the eigenvectors are orthonormal. An Eigenvalue Decomposition is a special kind of Jordan Decomposition in which the "Jordan form" matrix is diagonal. [7]
Application: LTI Systems with a state space representation. [8]

5 SVD

Let m and n be arbitrary ; we don’t not require $m \geqslant n$. Given M ϵ C^{mxn}, not necessarily of full rank, a SVD of M is a factorization

$$M = U\Sigma V^* \quad \longrightarrow \quad \begin{bmatrix} \uparrow & & \uparrow \\ m_1 & \dots & m_n \\ \downarrow & & \downarrow \end{bmatrix} = \begin{bmatrix} \uparrow & & \uparrow \\ u_1 & \dots & u_n \\ \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_p \end{bmatrix} \begin{bmatrix} \uparrow & & \uparrow \\ v_1 & \dots & v_n \\ \downarrow & & \downarrow \end{bmatrix}^{-T}$$

with $U \epsilon C^{mxn}$ unitary, $V \epsilon C^{nxn}$ unitary, $\Sigma \epsilon R^{mxn}$ diagonal.
In addition, it is assumed that the diagonal entries σ_j of Σ are non-negative and in non-increasing order; that is, $\sigma_1 \geqslant \sigma_2 \geqslant ... \geqslant \sigma_p \geqslant 0$ where p = min(m,n) The diagonal matrix Σ has the same shape as M even when M is not square, but U and V are always square unitary matrices. [9]

Different between Eigenvalue(Spectral) Decomposition and SVD:
$A \epsilon C^{mxn}, B \epsilon C^{mxm}$
$B = PDP^{-1}$; P is collection of eigenvectors of B ,D is diagonal matrix with eigenvalue
$A = U\Sigma V^*$; u is left eigenvector in m dimension, v is right eigenvector in n dimension , Σ is diagonal matrix with singular value.
$AA^* = U\Sigma V^*(U\Sigma V^*)^T = U\Sigma V^*(V\Sigma U^*) = U\Sigma(\Sigma U^*) = U\Sigma^2 U^*$ U is eigenvector of AA^*, Σ^2 is eigenvalue of AA^*

Application: Principal Components Analysis [9]

6 SCHUR

Consider this matrix:

$\text{A} = \begin{bmatrix} 13 & 8 & 8 \\ -1 & 7 & -2 \\ -1 & -2 & 7 \end{bmatrix}$ How do we get A^{60} ?

Easiest way is using $\text{A} = PDP^{-1} \qquad A^{60} = PD^{60}P^{-1}$ (eigenvalue decomposition), in fact that does not work for this matrix. However we can do something almost as good For any given matrix, the Schur decomposition allows to find another matrix that is similar to the given one and is upper triangular.

$A = UTU^*$ $\qquad U_2U_1AU_1U_2 = \begin{bmatrix} \lambda_1 & . & . \\ 0 & \lambda_2 & . \\ 0 & 0 & \lambda_3 \end{bmatrix} = U^*TU$

where T is upper triangular, with the eigenvalues of A on its diagonal, and U was a unitary matrix.
The Schur decomposition is not unique we choose an eigenvalue arbitrarily; as a consequence, there are different possible orderings of the eigenvalues of A on the main diagonal of T.
More in general, if $U^*AU = T$ is a Schur decomposition of A, we can take any unitary matrix Q such that Q^*TQ
is upper triangular, and use it to construct another Schur decomposition $(UQ)^*A(UQ) = Q^*TQ$ [10]

7 QR

QR factorization is often used to solve the linear least squares problem. Assume that M $\epsilon C^{mxn} (m \geq n)$ has full rank n.

$M = Q * R \qquad \longrightarrow \qquad \begin{bmatrix} \uparrow & & \uparrow \\ m_1 & \dots & m_n \\ \downarrow & & \downarrow \end{bmatrix} = \begin{bmatrix} \uparrow & & \uparrow \\ q_1 & \dots & q_n \\ \downarrow & & \downarrow \end{bmatrix} * \begin{bmatrix} r_{1,1} & \cdots & r_{1,n} \\ & \ddots & \vdots \\ & & r_{n,n} \end{bmatrix}$

Written out these equations as linear combination of $m_1, m_2, ...m_n$ and take the form;
$m_1 = r_{11} * q_1$
$m_2 = r_{12} * q_1 + r_{22} * q_2$
.
.
.
$m_n = r_{1n} * q_1 + r_{2n} * q_2 + ... + r_{nn} * q_n$
As a matrix formula, we have $M = \hat{Q} * \hat{R}$ called reduced QR Factorization
Q-factor:

- Q is mxn matrix with orthogonal columns $(Q^T * Q = I)$
- If M ıs a square (m=n) matrix, then Q is orthogonal $(Q^T * Q = Q * Q^T = I)$

R-factor:

- Q is nxn upper triangular matrix with non zero diagonal elements $(r_{kk} \neq 0)$
- R is non-singular (diagonal elements are non-zero)

Full QR Factorization: [11]
QR factorization of a matrix M ϵC^{mxn}, appending an additional m-n orthogonal to columns so $\hat{Q}$ becomes an mxn matrix.
2 Group: :
Orthogonal triangularization-Householder $Q_n...Q_2Q_1A = R$
Triangular orthogonalization- Gram-Schmidt $AR_1R_2...R_n = Q$

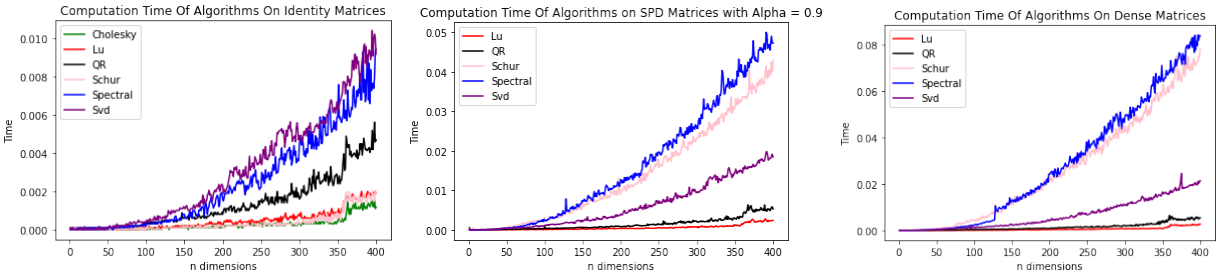
Algorithms for QR factorization
Modified Gram-Schmidt algorithm: complexity for mxn dimensional matrix;
$\sum_{k=1}^{n}(4m-1)(k-1)-3m) \quad = \quad (4m-1)\frac{n(n-1)}{2} + 3mn$
$= 2mn^2$ Flops

Householder Algorithm:
$\sum_{k=1}^{n} 4(m-k+1)(n-k+1) \quad = \quad \int_0^n 4(m-t)(n-t)\,dt$
$= 2mn^2 - \frac{2}{3}n^3$ Flops

SVD $\sim 2mn^2 + 11n^3$ flops
Normal equation $\sim mn^2 + \frac{1}{3}n^3$ flops

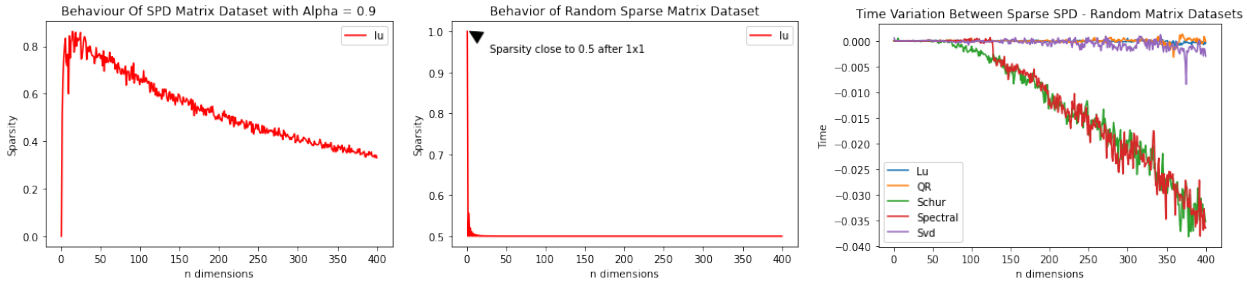
8 Results

All results can be reproduced from our project in **GitHub Repository** .



(a) (b) (c)

Figure 1. Illustrates the computation time of the algorithms versus increasing dimension of matrices. Since identity matrix has the eigenvalues on its diagonal it helps Schur and Spectral to compute fast than SVD however if there are other entries as it can be seen in (b) (c) those algorithms struggle to calculate eigenvalues and eigenvectors. In theorem SVD has the highest $2*m*n^2 + 11*n^3$ flops and Cholesky has the lowest $mn^2 + \frac{1}{3}n^3$ flops. As the result SVD also cost the highest time on compute and Cholesky is on the bottom of graph which cost the least time Specifically (a) shows that how all algorithms perform on identity matrix, naturally Symmetric Positive Definite, and (b) outlines 5 algorithms on SPD dataset however we had to leave out Cholesky because empirically since **Sklearn** uses randomization, Finally (c) illustrates the algorithms perform on dense dataset.



(a) (b) (c)

Figure 2. From (a) and (b) plots it can be seen that how the datasets behave when different sparsity rates are passed. As it is shown there is a non linearity situation in Symmetric Positive Definite matrix as the dimension increases. In (c) the main take away is how diagonalization effects on SPD and dense matrix; as the dimension of matrix increases SVD, QR, LU, decompositions are hardly effective however with Spectral and Schur, SPD matrix is decomposed easier

Algorithm	Outcome
QR-Householder	1.000000284809481
QR-Gram-Schmidt	0.981779891867633
Normal Equation	-0.390537644183687
Singular Value Decomposition	1.000000284809170
LUP	0.999999951596828

Table 1. Vandermonde Least Squares Problem Results : Thanks to the normalization, it is known that the true value of x (in A*x = b) will equal to 1. This means that all algorithms are accurate, especially SVD, QR-householder and LU with partial pivoting . QR-householder is more stable than QR using Gram-Schmidt that concludes the theorem in [12]

9 Conclusion

Given the guidance of the original paper on which this report is based on, we were able to explore the decompositional approach.
We confirm that the introduction of matrix decomposition into scientific computations brought convenient and sustainable platforms.
We have experienced how to do literature review and scripting pseudo codes to actual program via using open source tools. During the project it was quite easy to reconstruct or import modules for algorithms however while researching for dataset, especially Symmetric Positive Definite matrices, we could not really find a high quality open source dataset in Python Programming Language.
To sum up we have learned more insights and thought our fellow students the matrix factorization algorithms.

As a future work it can be possible to extend our project with some embedded factorization algorithms on different datasets.

10 Work Division

Muhammet USLU : I have reviewed research papers, state of the art applications and solved numerical examples on Cholesky, LU and Spectral Decompositions. I set up all the Jupyter Notebook structure to make Mao‘s test runs convenient, tidy and document the used algorithms to more readable. As we stated in this report some algorithms requires special type of matrices I did research for datasets. I have written LaTex format of the final report.

Mao PO : I was responsible for QR, Schur and Singular Value Decomposition. I wrote Matlab script for vandermonde matrix to test algorithms stability by solving Least Squares problems. I compared different matrices and algorithms from Muhammet‘s work and visualize by plotting graphics. I handle errors in Python program.

References

[1] G.W. Stewart. The decompositional approach to matrix computation. *Computing in Science Engineering*, 2(1):50–59, 2000.

[2] N.J.Higham. Cholesky factorization. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(2):251–254, 2009.

[3] Cederbaum Lorenz S. Vysotskiy, Victor P. On the cholesky decomposition for electron propagator methods: General aspects and application on c[sub 60]. *The Journal of Chemical Physics*, 132, 2010.

[4] Abdelrahim Zabadi, Ramiz Assaf, and Mohammad Kanan. Lu - decomposition computerized method to solve linear programming problems. 2017.

[5] Petya Vachranukunkiet Prawat Nagvajara Jeremy Johnson, Timothy Chagnon and Chika Nwankpa. Sparse lu decomposition using fpga.

[6] Sanjoy Dasgupta. Spectral methods. *CSE 291: Unsupervised learning*, spring 2008.

[7] Brian Winkel. Gustafson, g. b. *Differential Equations Course Materials*, September 2017.

[8] Munther Dahleh. Jordan decomposition.

[9] Luis M. Rocha Michael E. Wall, Andreas Rechtsteiner. Singular value decomposition and principal component analysis. 2003.

[10] L. Vandenberghe. Schur decomposition. spring 2020.

[11] L. Vandenberghe. Qr factorization. 2021.

[12] L.N. Trefethen and D. Bau. Numerical linear algebra. *Society for Industrial and Applied Mathematics*, 1997.