Homework 1 - Numerical Linear Algebra

Chase Wiederstein

December 2024

1 Executive Summary

This project involves the implementation and validation of the Schur algorithm for matrices with displacement ranks of 2 and 4. The objective is to compute the Cholesky factor R of a Toeplitz matrix T or asymmetric positive definite (SPD) matrix M, utilizing their displacement properties with respect to a downshift matrix Z. My code was completed in Python relying on *numpy* for basic computations and *scipy.linalg* for more advanced linear algebra computations. The project also explores the relationship between matrix structure and algorithmic performance, comparing Toeplitz and general SPD matrices to understand how displacement rank and problem formulation impact numerical accuracy and computational stability.

2 Rank 2 Displacement Validation

For the rank 2 displacement code, we were tasked with validating the code on both an SPD Toeplitz matrix T and a general SPD matrix M. For each matrix, I started with a normalized generator $G \in \mathbb{R}^{n\times 2}$ with displacements either $\nabla T = G\Phi G^T$ or $\nabla M = G\Phi G^T$ where $\Phi = (e_1 \quad -e_2) \in \mathbb{R}^2$. The eigenvalues of each matrix were also calculated to confirm its definiteness or to identify cases where the matrix was indefinite or semidefinite.

2.1 Toeplitz Matrices

I generated a total of three different Toeplitz matrices: SPD, symmetric indefinite, and an SPD with boosting. For each matrix, I also calculated its respective power series representation using:

$$T_{temp} = \sum_{j=0}^{n-1} Z^j \left(G\Phi G^T\right) \left(Z^j\right)^\top \tag{1}$$

By calculating $||T - T_{temp}||_2$, I could validate that the reconstructed matrix T_{temp} closely approximated the original matrix T confirming the correctness of its generator and displacement structure. I then calculated the true Cholesky named R_{true} using the *scipy.linalg* in order to calculate $||R - R_{true}||_2$ to validate the resulting Cholesky factor R from the Schur algorithm. I have also included flags in my Schur algorithm that are thrown when the hyperbolic rotation fails, pointing to issues with the definiteness of the matrix. Below I have included Table 1 which shows the r vectors that were created to test the algorithm and in Table 2, I have included the results.

	Vector
r_{dense}	$[125, 50, 40, 50, 30, 20, 20, 30, 30, 20]$
r_{boost}	$[125 + 100, 50, 40, 50, 30, 20, 20, 30, 30, 20]$
r_{sparse}	$[100, 0, 0, 0, 10, 0, 0, 0, 5, 0]$

Table 1: r vectors for Rank-2 Toeplitz

Matrix Type	$\kappa_2(T)$	$\|T - T_{\text{temp}}\|_2$	$\|R - R_{\text{real}}\|_2$
T_{base}	9.526484352282663	5.4848564076379104e-14	1.905469248165507e-13
T_{boost}	5.173828493477146	0.0	2.5982242526606805e-13
T_{sparse}	1.3258823646942466	0.0	4.263256414560601e-14

Table 2: Rank-2 Schur Algorithm Toeplitz Results

The rank-2 Schur algorithm demonstrates strong performance in accurately reconstructing T and computing the Cholesky factorization, even for matrices with moderate condition numbers. Sparse and boosted matrices offer additional stability and accuracy. These results confirm the reliability of the displacement-based Schur algorithm for Toeplitz matrices, with potential for extension to other structured matrix types which we will see later on.

2.2 General SPD Matrices

I then repeated a process similar to that in section 4.1 for various general SPD matrices M. For the errors, I used the following:

$$M_{temp} = \sum_{j=0}^{n-1} Z^j \left(G\Phi G^T\right) \left(Z^j\right)^\top \tag{2}$$

For each matrix, I calculated the errors: $||M - M_{temp}||_2$ and $||R - R_{true}||_2$. I have tried several different methods of generating a general SPD matrix. For my first matrix M_1, I generated a random matrix A of integers between 1 and 5, then set $M_1 = AA^\top + 100I$. For the next matrix, I used the same process as I did for M_1 except during the generation of A, I set all $a_{ij} = 0$ if $a_{ij} < 4$ to generate a sparse matrix M_2. For my last matrix, M_3, I created a sparse vector $r = [100, 0, 0, 1, 0, 0, 0, 1, 0, .01]$ to create a sparse Toeplitz matrix T. I then set $M_3 = TT^\top$ which will be SPD and reset r to be the first row. Below are the algorithms results for each matrix.

Matrix Type	$\kappa(T)$	$\|M - M_{\text{temp}}\|_2$	$\|R - R_{\text{real}}\|_2$
M_1	7.216034117685939	92.46830337026203	92.46830337026208
M_2	2.2035809294287088	79.60304606867693	79.60304606867683
M_3	1.0833858491927693	0.020100000001154517	0.020100000005209142

Table 3: Rank-2 Schur Algorithm General SPD Results

The SPD matrices used in these computations were designed to produce well-conditioned matrices, ensuring a fair evaluation of the Schur algorithm's stability while minimizing the influence of $\kappa(M)$. Despite having comparable condition numbers to the Toeplitz matrices, the Schur algorithm demonstrated significantly worse performance on these general SPD matrices, largely due to the lack of Toeplitz structure. The displacement structure $\nabla T = T - ZTZ^\top$ is better fit with the generator G for Toeplitz matrices, enabling better numerical stability. In contrast, the irregular structure of general SPD matrices leads to worse displacement representations and larger computational errors.

3 Rank 4 Displacement Validation

For the rank 4 displacement code, we were tasked with validating the code on a Toeplitz least squares problem $||b - Tx||_2$ by computing the Cholesky Factor of the normal equations matrix $T^\top T$ starting with a normalized generator $G \in \mathbb{R}^{n\times 2}$ and displacement $\nabla T = G\Phi G^\top$ with $\Phi = (e_1 \quad -e_2) \in \mathbb{R}^2$.

The process of this section is similar to what was seen in section 3. The Schur algorithm implemented in this project indirectly addresses the following least squares optimization problem:

$$\mathbf{x}_{\min} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{b} - T\mathbf{x}\|_2^2 \quad \Leftrightarrow \quad T^\top T\mathbf{x}_{\min} = T^\top \mathbf{b}. \tag{3}$$

This equivalence implies that solving the least squares problem (minimizing the residual norm) is equivalent to solving the system of normal equations, $T^\top T\mathbf{x} = T^\top \mathbf{b}$, where $T^\top T$ is the Gram matrix.

In the code, the Gram matrix $T^\top T$ is explicitly computed and then factorized using the Schur algorithm to obtain the upper triangular Cholesky factor R, such that

$$N = T^\top T = R^\top R$$

To validate this code, I have defined three errors: the displacement error defined as $||\nabla N - \nabla N_{\text{comp}}||_2$, the normal equation error as $||N - N_{\text{comp}}||_2$, and the Cholesky error as $||R - R_{\text{real}}||_2$. These errors ensure that the displacement-based Schur algorithm accurately computes the factorization. This validation confirms that the solutions obtained via both formulations ($T^\top T\mathbf{x} = T^\top \mathbf{b}$ or $\arg\min \|\mathbf{b} - T\mathbf{x}\|_2^2$) are consistent within the error margins analyzed in this project.

3.1 Rectangular Toeplitz

For testing, I have created three different matrices by creating $c \in \mathbb{R}^{10}$ and $r \in \mathbb{R}^4$ and then setting $T = \text{Toeplitz}(c, r) \in \mathbb{R}^{10\times 4}$. I have considered a dense matrix, a boosted dense matrix, and a sparse matrix for testing.

Matrix Type	$\kappa_2(N)$	$\|\nabla N - \nabla N_{\text{comp}}\|_2$	$\|N - N_{\text{comp}}\|_2$	$\|R - R_{\text{real}}\|_2$
T_{dense}	9.24094387950746	75.70125169012327	120.94023793946418	5.03004204270624
T_{boost}	1.2377175876970765	42.39543803315373	63.89004024884241	0.3720999324050239
T_{sparse}	1.2200909810259835	0.9853306987648812	0.9901960784313478	0.049866145627454485

Table 4: Rectangular Rank-4 Schur Least Squares Algorithm Results

When calculating the Gram matrix, the Toeplitz structure will be annihalted resulting in an SPD Gram matrix. This lack of inherent structure explains the larger errors compared to the rank-2 Toeplitz results. However, this rank 4 problem produces better results than for the rank 2 on a general SPD. This may be due to the least squares framework which helps mitigate the error accumulation when working with the SPD Gram matrix. The higher rank of 4 may also play into this as it may produce higher accuracy improving the final Cholesky factorization, but more testing would need to be done to confirm this. Overall, this algorithm is seen to perform well and will work best with a better conditioned matrix $N = T^\top T$.

3.2 Square Toeplitz

I also tested the square cases for $c \in \mathbb{R}^5$ and $r \in \mathbb{R}^5$. A similar process was used as to the rectangular cases from above. The table below shows the results.

Matrix Type	$\kappa_2(N)$	$\|\nabla N - \nabla N_{\text{comp}}\|_2$	$\|N - N_{\text{comp}}\|_2$	$\|R - R_{\text{real}}\|_2$
T_{dense}	10.259354231915296	130.2741577424935	200.365684245755742	6.075023169144524
T_{boost}	1.3783727956719745	99.59731480150477	146.31125893980558	0.7904475997488709
T_{sparse}	2.005693071773367	1.0	0.9999999999999929	0.12216797124063349

Table 5: Square Rank-4 Schur Least Squares Algorithm Results

As compared to the rectangular cases, the square Toeplitz matrices show slightly higher errors, even with similar matrix conditioning. This may be due to the difficulty in maintaining the displacement structure in square matrices. Both setups consistently exhibit trends where dense matrices have the largest errors, boosted matrices improve their performance, and sparse matrices provide the most accurate results. These findings reaffirm the rank-4 Schur algorithm's reliance on well-conditioned matrices and highlight the importance of structure and algorithmic precision.

4 Conclusion

In this project, the performance of the Schur algorithm was validated across both rank-2 and rank-4 displacement settings. For Toeplitz matrices, the rank-2 algorithm demonstrated minimal errors due to its

compatibility with the matrix's predictable structure, while for general SPD matrices, the rank-2 algorithm underperformed. The rank-4 algorithm showed increased errors when the Toeplitz structure was lost in the Gram matrix $N = T^\top T$. When comparing the rectangular and square Toeplitz under the rank-4 algorithm, the rectangular outperformed the square matrix when both were given similar matrices. However, the rank-4 algorithm performed significantly better than the rank-2 algorithm on general SPD matrices, aided by its use of least squares to compute the Cholesky factorization. Overall, the Schur algorithm worked best when able to exploit a Toeplitz structure and all Algorithms were shown to have success in solving for the Cholesky factorization.