Fast Iterative Solvers
Project 1: Krylov Subspace Methods

Samar Bahman
416255

June 6, 2023

GMRES Method

Full GMRES Method

For full GMRES algorithm $m = 600$ was chosen as the restart parameter. The relative residual is plotted on a semi-scale plot against the iteration index as shown in Figure (1). As shown, the Guass-Seidel preconditioning reaches the convergence criteria of $||r_k||_2/||r_0||_2 = 10^{-8}$ in fewer iterations.

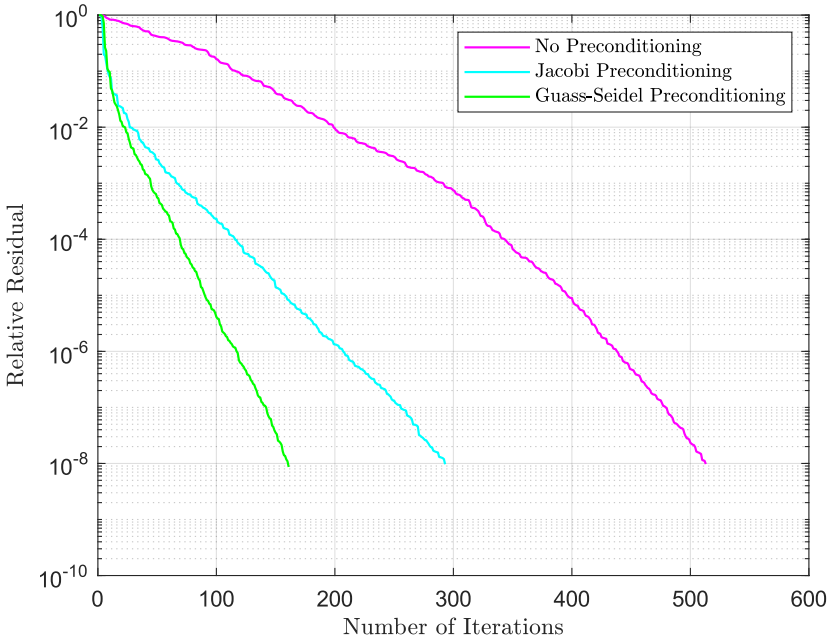

Figure 1: Relative residual against the iteration index

The number of Krylov vectors to solve the problem with and without preconditioning is shown in Table (1).

Table 1: Number of Krylov vectors for full GMRES algorithm without restart

Method	**Number of Krylov Vectors**
No Preconditioning	512
Jacobi Preconditioner	293
Gauess-Seidel Preconditioner	161

Restarted GMRES Method

The comparison of runtime for different restart parameters and full GMRES is shown in Table (2). As shown, the restarted GMRES is not significantly faster than the full GMRES algorithm.

Table 2: Runtime [s]

Method	**m = 30**	**m = 50**	**m = 100**	**m = 600**
No Preconditioning	10.41	7.47	5.73	3.06
Jacobi Preconditioner	1.47	1.44	1.57	1.36
Gauess-Seidel Preconditioner	0.88	0.7	0.74	0.68

In principle, by reducing the restart parameter 'm' and using the result of previous iteration as the guess solution for the next iteration, total time of convergence reduces. Since MATLAB uses double-precision floating-point arithmetic by default, a more accurate representation of the numbers are involved in the algorithm which leads to more storage capacity allocation and longer run time of the algorithm. The number of restart cycles is shown in Table (3) for every preconditioner and restart parameter.

Table 3: Number of restart cycles for GMRES algorithm

Method	**m = 30**	**m = 50**	**m = 100**
No Preconditioning	211	53	17
Jacobi Preconditioner	15	8	5
Gauess-Seidel Preconditioner	9	5	3

In the full GMRES algorithm, the Krylov subspace grows with each iteration which requires more memory to store the increasingly large number of basis vectors. However, restarted GMRES limits the size of the Krylov subspace by periodically restarting the iterations. Using a smaller Hessenberg matrix and fewer Krylov vectors facilitates the computation. By creating a smaller Krylov subspace, restarted GMRES significantly reduces the memory requirements and usage.

Orthogonality

The orthogonality of the Krylov vectors is shown if Figure (2) for full GMRES algorithm (without preconditioning).

As it can be seen, the dot product of the first vector in the set of Krylov vectors with the subsequent vectors is increasing. This indicates a loss of orthogonality which may arise due to several reasons, such as round-off errors, ill-conditioning, and numerical instability due to large condition numbers. Using techniques such as restarting and preconditioning mitigates the loss of orthogonality.

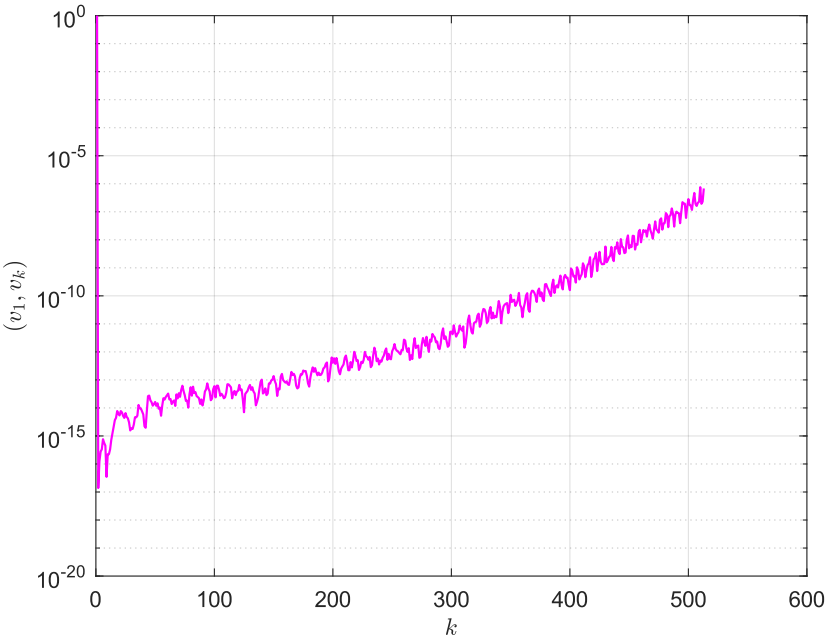

Figure 2: Orthogonality of Krylov Vectors for full GMRES algorithm

Conjugate Gradient Method

The conjugate gradient method was implemented without preconditioning to reach the convergence criteria of $||r_k||_2/||r_0||_2 = 10^{-8}$. The error in A-norm, i.e., $||e||_A = \sqrt{(Ae, e)}$, as well as the residual in standard 2-Norm, i.e., $||r||_2 = \sqrt{(r, r)}$, against iteration index is plotted on a semi-log scale shown in Figure (3).

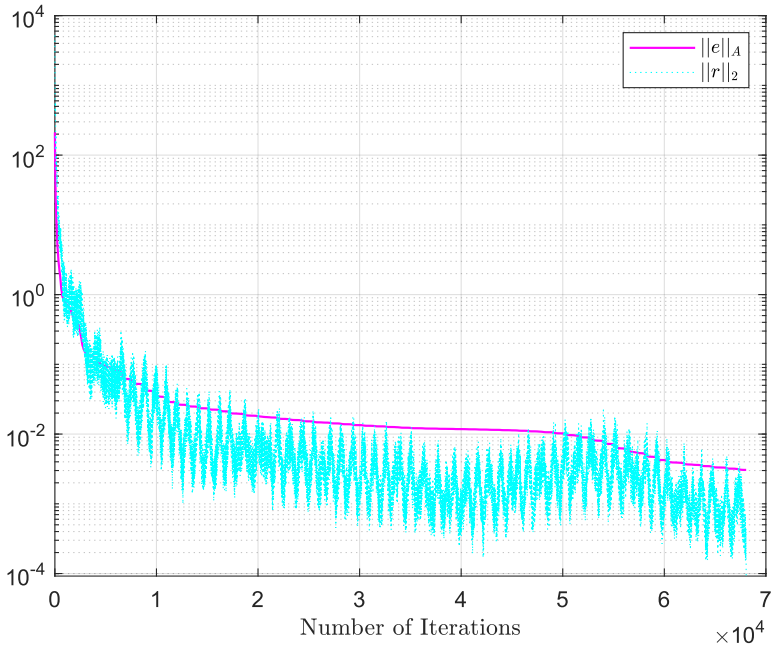

Figure 3: Comparison of error in A-norm and residual in 2-Norm against iteration index

As shown in Figure (3), the error in A-norm converges monotonically while the residual in the standard 2-Norm oscillates until convergence. According to Theorem 0.1, error in the A-norm decreases in each iteration which describes the monotonically decreasing behaviour. However, the residual in the standard 2-Norm shown oscillates since it is not bound.

Theorem 0.1 *Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite, and let $x = A^{-1}b$, for arbitrary $b \in \mathbb{R}^n$. Then for the m^{th} iterate of the CG Algorithm*

$$||x_m - x||_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m ||x_0 - x||_A$$

where $\kappa = \lambda_{max}(A)/\lambda_{min}(A)$.

Several other reasons could exist, namely:

1. Ill-conditioned matrix: The matrix used for the conjugate gradient method maybe ill-conditioned. The large condition number of the ill-conditioned matrix causes numerical instability. Since the conjugate gradient method is sensitive to the conditioning of the matrix, oscillatory behaviour of the residual may arise.

2. Round-off errors: Round-off errors can have a more significant impact on the residual in comparison with the A-norm error in each iteration.

3. Convergence criteria: Using a smaller tolerance or increasing the maximum number of iterations may result in a less oscillatory convergence of the residual norm.

4. Residual computation: Using a different norm or approximation of the true solution may help mitigate the oscillatory behavior.

5. Preconditioning: The conjugate gradient method is very sensitive to the preconditioning. Adjusting the preconditioning technique or using a different preconditioner smooths the oscillatory behavior.

Since MATLAB uses double-precision floating-point arithmetic by default, a more accurate representation of the numbers are involved in the algorithm which leads to convergence in less iteration numbers. However, this high-precision numbers require more storage capacity which results longer run time of the algorithm.