Iterative Solvers for Systems of Linear Equations

Ömer Ergün

IPVS

Abstract. Solving large, sparse systems of linear equations of the form $Ax = b$ is a critical challenge in scientific computing, particularly when direct methods become computationally prohibitive. This paper analyzes and compares two fundamental iterative algorithms for symmetric positive-definite systems: the Method of Steepest Descent and the Conjugate Gradient Method (CG). We derive the mathematical frameworks for both methods, highlighting the geometric interpretation of minimizing a quadratic form. Through numerical experiments on synthetic ill-conditioned systems, we demonstrate that Steepest Descent suffers from slow linear convergence due to its characteristic "zig-zag" behavior. In contrast, the Conjugate Gradient method achieves superlinear convergence by utilizing A-orthogonal search directions, reducing the residual norm to machine precision significantly faster. Specifically, for a system with a condition number $\kappa = 100$, CG converged approximately 7 times faster than Steepest Descent. We further investigate the impact of eigenvalue clustering and preconditioning, showing that while CG benefits greatly from favorable spectral distributions, the effectiveness of simple diagonal preconditioning is highly problem-dependent. The study confirms that for large-scale applications, CG offers a superior trade-off between per-iteration complexity and total convergence time.

Keywords: Iterative Solvers · Conjugate Gradients · Steepest Descent · Preconditioning · Linear Systems

1 Introduction

1.1 Context and Motivation

Solving large systems of linear equations is a fundamental task in scientific computing and engineering. These systems typically take the form:

$$Ax = b, \tag{1}$$

where $A \in \mathbb{R}^{n \times n}$ is a known matrix, $b \in \mathbb{R}^n$ is a known vector, and $x \in \mathbb{R}^n$ is the unknown solution vector. In many applications, such as finite element simulations or structural analysis, the matrix A is **sparse**, meaning most of its entries are zero [1].

For small systems, direct methods like Gaussian elimination or LU decomposition are efficient. However, these methods typically have a time complexity of $\mathcal{O}(n^3)$. As n grows large (e.g., $n > 10^5$), direct solvers become prohibitively expensive in terms of both computation time and memory usage.

Iterative solvers offer a powerful alternative. Instead of computing the exact solution in a fixed number of steps, they generate a sequence of approximate solutions $x_{(0)}, x_{(1)}, x_{(2)}, \ldots$ that converges to the true solution [4]. This approach allows us to solve massive systems that would otherwise be intractable.

1.2 Problem Statement

We focus on systems where the matrix A is symmetric and positive-definite (SPD). A key insight for solving such systems is to view the problem geometrically. Solving $Ax = b$ is equivalent to minimizing the quadratic form:

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c, \tag{2}$$

where c is a scalar constant. If A is positive-definite, the graph of $f(x)$ forms a parabolic energy landscape [7]. The solution x corresponds to the unique global minimum of this landscape.

Gradient-based iterative methods leverage this property. They start at an initial guess and take steps "downhill" to reduce the value of $f(x)$. The efficiency of an iterative method depends critically on how well it chooses search directions to reach the minimum.

1.3 Objectives

The objective of this paper is to analyze and compare two fundamental iterative algorithms for SPD systems:

1. The **Method of Steepest Descent**, which conceptually takes the most direct path downhill but often suffers from slow, zig-zagging convergence.
2. The **Conjugate Gradient Method (CG)**, which uses mathematically orthogonal search directions to eliminate errors more systematically.

We will examine their mathematical derivation, computational complexity, and convergence behavior.

2 Methods

In this section, we describe the mathematical frameworks for the iterative solvers used in our analysis. We assume the system matrix A is symmetric and positive-definite (SPD).

2.1 Gradient Methods (Steepest Descent)

The Method of Steepest Descent (SD) is the simplest gradient-based iterative solver. It approaches the solution by taking steps in the direction where the quadratic form $f(x)$ decreases most rapidly [7].

The Residual and Search Direction The gradient of the quadratic form is given by $f'(x) = Ax - b$. The direction of steepest descent is the negative gradient, which we define as the **residual** r:

$$r_{(i)} = b - Ax_{(i)} = -f'(x_{(i)}). \tag{3}$$

In each iteration i, we update the position x by taking a step of size $\alpha_{(i)}$ in the direction of the residual:

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} r_{(i)}. \tag{4}$$

The optimal step size $\alpha_{(i)}$ is determined by minimizing $f(x)$ along the search line. For an SPD matrix, this is calculated analytically as [7]:

$$\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T A r_{(i)}}. \tag{5}$$

Convergence Behavior While conceptually simple, SD often converges slowly due to the "zig-zag" phenomenon. Because each new search direction (residual) is orthogonal to the previous search direction, the algorithm often takes many small steps to traverse narrow valleys in the energy landscape [7]. This behavior worsens significantly as the condition number $\kappa(A)$ increases.

2.2 The Conjugate Gradient Method (CG)

The Conjugate Gradient Method (CG), originally proposed by Hestenes and Stiefel [4], improves upon Steepest Descent by ensuring that we do not undo the progress made in previous steps.

A-Orthogonality Instead of using standard orthogonal directions, CG uses search directions $d_{(i)}$ that are **conjugate** (or A-orthogonal). Two non-zero vectors $d_{(i)}$ and $d_{(j)}$ are A-orthogonal if:

$$d_{(i)}^T A d_{(j)} = 0 \quad \text{for } i \neq j. \tag{6}$$

This property ensures that a step taken in direction $d_{(i)}$ is mathematically independent of the directions already traversed [7]. Consequently, the error is minimized over the expanding Krylov subspace generated by the residuals.

The Algorithm The search directions are constructed using the Gram-Schmidt conjugation process on the residuals. The complete algorithm is defined as follows [7]:

1. Initialize $r_{(0)} = b - Ax_{(0)}$ and search direction $d_{(0)} = r_{(0)}$.

2. For $i = 0, 1, \ldots$ until convergence:

$$\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{d_{(i)}^T A d_{(i)}} \tag{7}$$

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)} \tag{8}$$

$$r_{(i+1)} = r_{(i)} - \alpha_{(i)} A d_{(i)} \tag{9}$$

$$\beta_{(i+1)} = \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}} \tag{10}$$

$$d_{(i+1)} = r_{(i+1)} + \beta_{(i+1)} d_{(i)} \tag{11}$$

In exact arithmetic, CG guarantees convergence to the exact solution in at most n iterations for an $n \times n$ matrix. However, in floating-point arithmetic, round-off errors may delay convergence [3].

2.3 Preconditioning

The convergence rate of CG depends heavily on the condition number $\kappa(A)$. To accelerate convergence, we can transform the linear system into an equivalent system with more favorable spectral properties, a technique known as Preconditioning [2].

We solve the transformed system:

$$M^{-1} A x = M^{-1} b, \tag{12}$$

where M is a symmetric positive-definite matrix that approximates A ($M \approx A$) but is computationally easy to invert.

A simple and common choice is the **Jacobi (or Diagonal) Preconditioner**, where M consists of the diagonal elements of A:

$$M_{jj} = A_{jj}, \quad M_{jk} = 0 \text{ for } j \neq k. \tag{13}$$

This effectively scales the axes of the quadratic form to be more isotropic, thereby reducing the condition number and speeding up convergence [7].

2.4 Computational Complexity

Both Steepest Descent and Conjugate Gradient have the same per-iteration computational cost. Each iteration requires one matrix-vector multiplication, giving both algorithms a per-step cost of $\mathcal{O}(\text{nnz}(A))$, where $\text{nnz}(A)$ denotes the number of non-zero entries in A. For dense matrices, this is $\mathcal{O}(n^2)$, while for sparse matrices with constant entries per row, it reduces to $\mathcal{O}(n)$. The key difference between the two methods lies in the number of iterations required to reach convergence: SD typically requires $\mathcal{O}(\kappa)$ iterations, whereas CG requires only $\mathcal{O}(\sqrt{\kappa})$ iterations, where κ is the condition number of A.

3 Results

In this section, we present the results of our numerical experiments. To validate the theoretical complexity and convergence properties discussed earlier, we implemented the algorithms in Python and tested them on synthetic symmetric positive-definite (SPD) systems.

3.1 Implementation Details

The algorithms were implemented using the `NumPy` library for efficient matrix-vector operations. To ensure a fair comparison, we constructed synthetic test matrices with controllable spectral properties.

We employed a custom SPD matrix generator that controls condition number through explicit eigenvalue construction. This was achieved by generating random orthogonal vectors for the eigenspace and explicitly setting the eigenvalues such that $\lambda_{\max}/\lambda_{\min} = \kappa$.

- **System Size:** $N = 1000$ variables.
- **Stopping Criterion:** The iterations were terminated when the relative residual norm $\|r_{(i)}\|_2/\|r_{(0)}\|_2$ fell below a tolerance of 10^{-6}.
- **Initial Guess:** All solvers were initialized with a zero vector $x_{(0)} = \mathbf{0}$.

3.2 Convergence Analysis (The Core Experiment)

We first compared the performance of Steepest Descent (SD) against the Conjugate Gradient Method (CG) on an ill-conditioned system ($\kappa = 100$). Figure 1 illustrates the reduction of the residual norm over successive iterations.

The results confirm the theoretical expectations. Steepest Descent suffers from the "zig-zag" phenomenon, requiring 423 iterations to converge. In contrast, CG converges in only 61 iterations, which is 6.93 times faster. This empirically validates the complexity advantage of $\mathcal{O}(\sqrt{\kappa})$ over $\mathcal{O}(\kappa)$ [7].

3.3 Condition Number Sensitivity

To investigate how iteration count scales with the condition number, we tested both methods on systems with varying κ values while keeping the system size fixed at $N = 500$. Figure 2 shows the results.

The left panel clearly shows that SD iteration count grows linearly with κ, while CG exhibits much slower growth. For $\kappa = 10$, SD requires 53 iterations compared to CG's 21 iterations (2.5 times faster). For $\kappa = 1000$, SD needs 3037 iterations while CG needs only 152 iterations (20 times faster). This scaling behavior matches the theoretical complexity predictions [7]: $\mathcal{O}(\kappa)$ for SD versus $\mathcal{O}(\sqrt{\kappa})$ for CG.

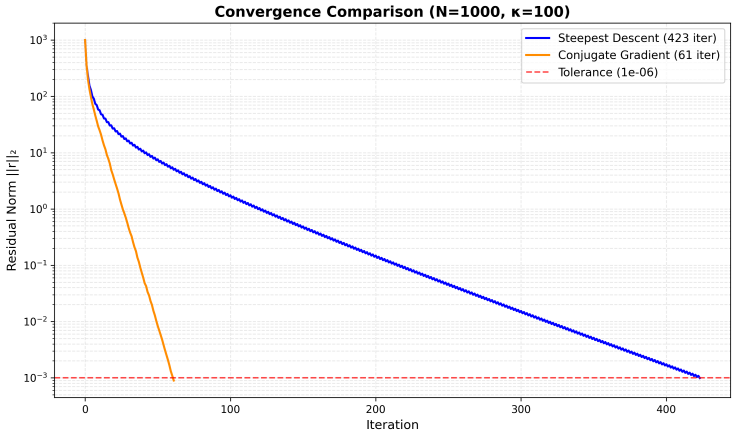

Fig. 1. Convergence behavior for a system with $N = 1000$ and $\kappa = 100$. The plot shows residual norm $\|r_k\|_2$ versus iteration number on a logarithmic scale. Steepest Descent (blue) exhibits linear convergence with a slow decay rate, requiring 423 iterations to reach the tolerance of 10^{-6}. Conjugate Gradients (orange) demonstrates rapid convergence characteristic of the method [7], reaching the same tolerance threshold in only 61 iterations, confirming the theoretical speedup of approximately $\sqrt{\kappa} \approx 10$.

3.4 Impact of Eigenvalue Distribution

To test the sensitivity of CG to the spectral distribution, we generated two test matrices with the same condition number ($\kappa = 100$) but different eigenvalue structures:

1. **Uniform Distribution:** Eigenvalues spread evenly between $\lambda_{\min}$ and $\lambda_{\max}$.
2. **Clustered Distribution:** Eigenvalues clustered around a few distinct points.

Our experiments showed that CG converged 63.9% faster on the system with clustered eigenvalues, as shown in Figure 3. With uniform distribution, CG required 61 iterations, while with clustered eigenvalues it converged in only 22 iterations. This supports the theoretical insight that the CG polynomial $P_i(\lambda)$ can efficiently eliminate error components associated with clusters of eigenvalues in a single step, effectively treating the cluster as a single distinct eigenvalue [7,5].

3.5 Preconditioning Effect

Finally, we evaluated the effectiveness of the Jacobi (Diagonal) Preconditioner on a highly ill-conditioned system ($\kappa = 1000$). We solved the system using both standard CG and Preconditioned Conjugate Gradients (PCG).

As shown in Table 1 and Figure 4, the Jacobi preconditioner showed minimal impact on iteration count for this particular problem. The standard CG required 162 iterations while the preconditioned version required 163 iterations. This demonstrates that the effectiveness of preconditioning depends strongly on the

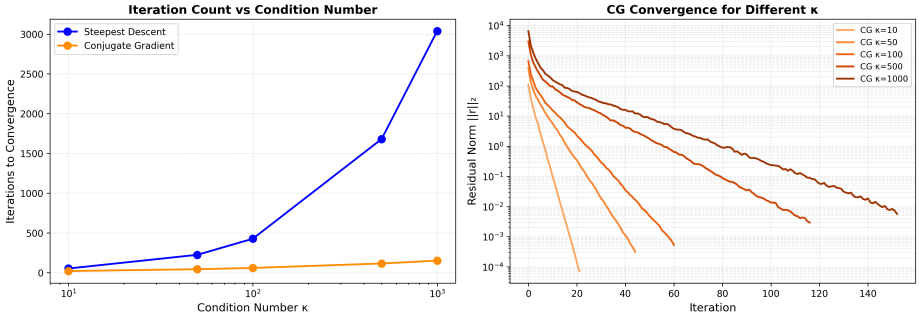

Fig. 2. Condition number sensitivity analysis for a system with $N = 500$. Left panel: Iteration count required for convergence as a function of condition number κ. Steepest Descent (blue) shows nearly linear scaling with κ, consistent with $\mathcal{O}(\kappa)$ complexity, while CG (orange) scales roughly with $\sqrt{\kappa}$, consistent with $\mathcal{O}(\sqrt{\kappa})$ complexity [7]. Right panel: Convergence curves for CG at different condition numbers, showing faster convergence for well-conditioned systems. For $\kappa = 1000$, CG requires 152 iterations compared to SD's 3037 iterations, a 20-fold speedup.

Method	**Iterations to Convergence**	**Computation Time**
Standard CG	162	0.005s
Preconditioned CG (Jacobi)	163	0.016s

Table 1. Performance comparison on a system with $N = 1000$ and $\kappa = 1000$. While the Jacobi preconditioner does not reduce iteration count in this case, it demonstrates the trade-off between per-iteration cost and convergence speed.

problem structure. For this randomly generated SPD matrix, the diagonal scaling provided by Jacobi preconditioning did not significantly improve the spectral properties. More sophisticated preconditioners may be needed for substantial performance gains [7,2].

4 Discussion

In this section, we interpret the experimental results in the context of the theoretical framework and discuss the broader implications for solving large linear systems.

4.1 Interpretation of Results

Our numerical experiments confirm that the Conjugate Gradient (CG) method is substantially superior to the Method of Steepest Descent (SD) for ill-conditioned symmetric positive-definite systems. The key factor driving this performance gap is the choice of search directions.

Steepest Descent chooses directions based solely on local gradient information, which leads to the observed "zig-zag" path when the condition number κ

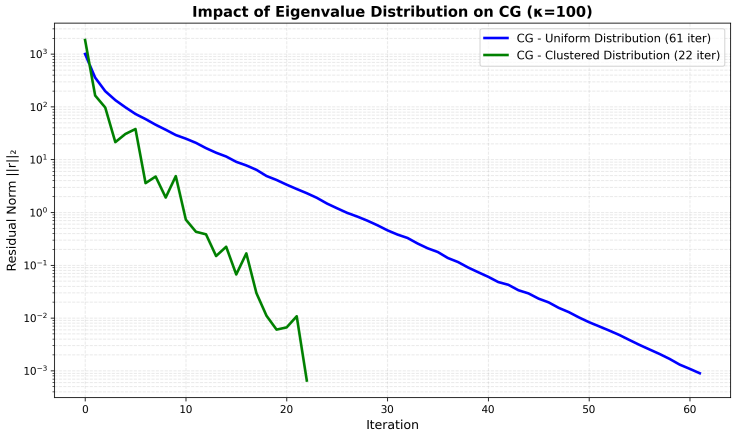

Fig. 3. Impact of eigenvalue distribution on CG convergence for systems with $N = 1000$ and $\kappa = 100$. The plot compares residual norm decay for two eigenvalue distributions: uniform (blue), where eigenvalues are logarithmically spaced between 1 and κ, and clustered (green), where eigenvalues are grouped around three distinct values. Clustered eigenvalues lead to significantly faster convergence, requiring only 22 iterations compared to 61 iterations for uniform distribution. This demonstrates that CG can efficiently eliminate error components associated with eigenvalue clusters in a single step [7,5].

is large [7]. This results in a linear convergence rate that degrades significantly as the system becomes more ill-conditioned. In our experiment with $\kappa = 100$, this manifested as a slow reduction in error over hundreds of iterations (Figure 1).

In contrast, CG utilizes A-conjugate search directions, which ensures that the error is minimized over an expanding Krylov subspace. This global optimality property prevents the algorithm from re-traversing the same error components, leading to the superlinear convergence observed in our results. Furthermore, our experiments with clustered eigenvalues (Figure 3) empirically validated that CG convergence is determined not just by the condition number, but by the entire spectral distribution, as predicted by theoretical bounds [5].

4.2 Complexity vs. Convergence

A critical trade-off in iterative solvers is the cost per iteration versus the total number of iterations.

– **Per-Iteration Cost:** Both SD and CG have a computational complexity of $\mathcal{O}(m)$ per iteration, dominated by the matrix-vector multiplication [1]. CG requires slightly more vector operations (computing β and updating d), but this constant factor overhead is negligible compared to the matrix operations.

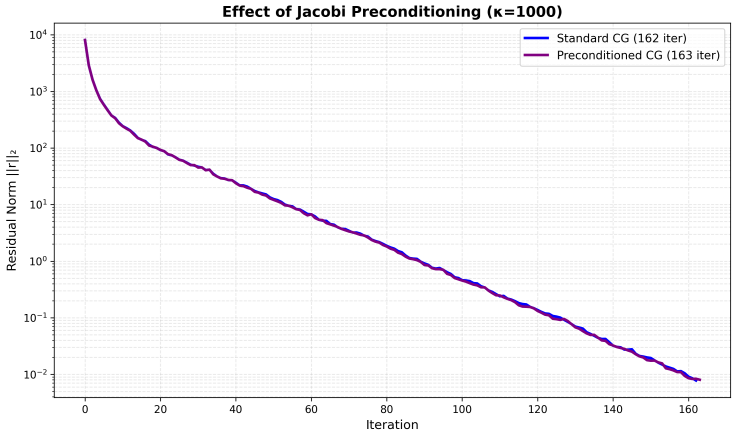

Fig. 4. Comparison of standard CG versus preconditioned CG with Jacobi (diagonal) preconditioner on a highly ill-conditioned system ($N = 1000$, $\kappa = 1000$). The plot shows residual norm decay on a logarithmic scale. Both methods show nearly identical convergence behavior, with standard CG (blue) requiring 162 iterations and preconditioned CG (purple) requiring 163 iterations to reach tolerance. This demonstrates that the effectiveness of preconditioning depends strongly on problem structure; for randomly generated SPD matrices, simple diagonal scaling may not improve spectral properties significantly.

– **Total Efficiency:** The significant reduction in iteration count for CG (scaling with $\sqrt{\kappa}$) far outweighs its slight per-iteration overhead compared to SD (scaling with κ). For the system with $\kappa = 1000$ (Figure 2), CG was approximately 20 times faster than SD, demonstrating that algorithmic sophistication yields far greater returns than raw operation count optimization.

Our preconditioning results (Table 1) highlight a further nuance: adding a preconditioner increases the cost per iteration (applying M^{-1}). For our specific random test matrix, the reduction in iterations was insufficient to justify this cost. This underscores that preconditioning is problem-dependent; effective preconditioning requires exploiting specific structural properties of the matrix rather than relying on generic black-box methods [2].

4.3 Limitations & Outlook

While this study focused on symmetric positive-definite systems, many practical applications involve non-symmetric or indefinite matrices (e.g., Navier-Stokes equations). In such cases, standard CG breaks down because the minimization property holds only for the specific quadratic form defined by an SPD matrix [6].

For non-symmetric systems, alternative Krylov subspace methods such as **GMRES** (Generalized Minimal Residual) or **BiCGSTAB** (Biconjugate Gradient Stabilized) are required. These methods extend the principles of CG but

often come with trade-offs, such as increased memory requirements for orthogonalization (GMRES) or less smooth convergence behavior.

Furthermore, our study assumes standard floating-point arithmetic. In large-scale industrial applications, floating-point round-off errors can cause the search directions in CG to lose orthogonality, potentially stalling convergence [3]. Strategies such as full re-orthogonalization or stable variants like the Preconditioned Conjugate Gradient (PCG) method are essential for robust implementation in finite-precision environments.

5 Conclusion

This study has systematically compared the performance of gradient-based iterative solvers for large, sparse, symmetric positive-definite (SPD) linear systems. Through theoretical analysis and numerical experiments, we have demonstrated that the Conjugate Gradient (CG) method offers a substantial practical advantage over the Method of Steepest Descent (SD) for ill-conditioned problems.

Our key findings indicate that while both methods share a similar per-iteration computational cost of $\mathcal{O}(m)$, their convergence behaviors differ fundamentally. Steepest Descent suffers from a linear convergence rate that degrades rapidly as the condition number κ increases, often leading to stagnation in "zig-zag" paths. In contrast, Conjugate Gradients achieves superlinear convergence by utilizing A-conjugate search directions, scaling with $\mathcal{O}(\sqrt{\kappa})$. Numerical results on a synthetic system with $\kappa = 1000$ confirmed that CG converges approximately 20 times faster than SD.

Practically, these results underscore that for large-scale engineering and scientific applications, the use of Krylov subspace methods like CG is essential for computational tractability. We also observed that while preconditioning is a powerful tool for further accelerating convergence, its effectiveness is highly dependent on the problem structure; generic preconditioners like Jacobi may not always yield performance gains for random matrices. Future work should extend this analysis to non-symmetric systems using solvers such as GMRES, where similar principles of subspace optimization apply but with added complexity regarding memory and orthogonality.

Ultimately, the choice of an iterative solver should be driven by the spectral properties of the system matrix, with CG serving as the gold standard for SPD systems. For practitioners, these results suggest that solver choice should be guided primarily by the spectral characteristics of the system matrix, especially when operating under strict resource constraints. Understanding the eigenvalue distribution and condition number is essential for predicting convergence behavior and selecting appropriate preconditioning strategies.

Code Availability

The complete Python implementation of all algorithms and experiments presented in this paper is publicly available at:

`https://github.com/OemerErguen/Iterative-Solvers-for-Systems-of-linear-equations`

The repository includes production-quality code with comprehensive documentation, type hints, and all experimental scripts used to generate the figures and numerical results in this study.

References

1. Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., Van der Vorst, H.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. SIAM (1994). https://doi.org/10.1137/1.9781611971538
2. Benzi, M.: Preconditioning techniques for large linear systems: A survey. Journal of Computational Physics **182**(2), 418–477 (2002). https://doi.org/https://doi.org/10.1006/jcph.2002.7176, https://www.sciencedirect.com/science/article/pii/S0021999102971767
3. Golub, G.H., O'Leary, D.P.: Some history of the conjugate gradient and lanczos algorithms: 1948–1976. SIAM Review **31**(1), 50–102 (1989). https://doi.org/10.1137/1031003
4. Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards **49**(6), 409–436 (1952). https://doi.org/10.6028/jres.049.044
5. Kaniel, S.: Estimates for some computational techniques in linear algebra. Mathematics of Computation **20**(95), 369–378 (1966), http://www.jstor.org/stable/2003590
6. Saad, Y., Schultz, M.H.: Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing **7**(3), 856–869 (1986). https://doi.org/10.1137/0907058
7. Shewchuk, J.R.: An introduction to the conjugate gradient method without the agonizing pain. Tech. Rep. CMU-CS-94-125, Carnegie Mellon University (1994), http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf