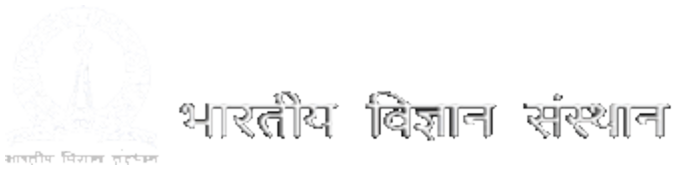

Numerical Analysis of Some Preconditioners and Associated Error Estimators for Solving Linear Systems

Journal:	*IISc - Masters Thesis Processing*
Thesis ID	masters-2020-0038.R1
Manuscript Type:	Synopsis and Thesis
Date Submitted by the Author:	24-Sep-2020
Complete List of Authors:	Dugar, Abhishek; IISc, Computational & Data Sciences(CDS)
Keywords:	

Numerical Analysis of Some Preconditioners and Associated Error Estimators for Solving Linear Systems

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

Master of Technology

IN

Faculty of Engineering

BY

Abhishek Dugar

Department of Computational and Data Sciences
Indian Institute of Science
Bangalore – 560 012 (INDIA)

September, 2020

Declaration of Originality

I, **Abhishek Dugar**, with SR No. **06-02-00-10-22-17-1-15051** hereby declare that the material presented in the thesis titled

Numerical Analysis of Some Preconditioners and Associated Error Estimators for Solving Linear Systems

represents original work carried out by me in the **Department of Computational and Data Sciences** at **Indian Institute of Science** during the years **2017-20**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Murugesan Venkatapathi Advisor Signature

DEDICATED TO

My family and friends

who have been constantly supporting me

Acknowledgements

I would like to thank my guide, Dr. Murugesan Venkatapathi for letting me be a part of his lab and giving me the opportunity to work on this topic. He has been a constant support since the beginning and showed immense patience throughout the lifetime of my work. The thesis would not have been possible without his guidance and expertise.

I would also like thank my peers at CDS who have provided support, both emotional and academic. There was never a gloomy day at CDS because of them.

Finally, I would like to thank my family who have always been there for me for any obstacles I faced. They provided me with the confidence to follow my dreams.

Abstract

Convergence of iterative algorithms in solving large linear systems is largely affected by the condition number of the matrix. Preconditioners reduce the condition number of the system matrix, thereby letting the linear system converge in fewer iterations. First, we perform a theoretical study on the expected iterations saved due to a general purpose preconditioner as a function of matrix size, tolerance, condition number and the linear solver (CG or GMRES). A metric is suggested for evaluating gains with respect to the iterations required in preconditioned and non-preconditioned systems, and experimental analysis of the same will be presented. These experiments explore split Jacobi and Incomplete Cholesky preconditioners for symmetric positive definite (SPD) matrices. The second part of this work focuses on the role of error estimators in realizing the gains of a preconditioner. We apply error estimators for non-preconditioned and preconditioned solvers and compare their significance in both cases.

Contents

CONTENTS

List of Figures

LIST OF FIGURES

Chapter 1

Introduction

Preconditioners have become an essential tool in speeding up linear solvers. Potentially we stand to gain a lot by using these, as we will see in Figure 1.1. But section 1.5.1 highlights that using them might turn out to be counter-productive. There might be cases where the overhead of using a preconditioner might not justify the iterations saved and also cases where it might make the system harder to solve. This motivates us to revisit and study the effect of the preconditioners on the the condition number of the system matrix, and the actual number of iterations saved, especially as an experimental analysis.

Before we address that, we define key concepts that we will be using going forward and section 1.6 lists our objectives. Then in Chapter 2, we try to answer the above question for at least a subset of the matrices.

1.1 Condition Number

In general a problem can be written as a function $f : X \to Y$ [27] (here X and Y are normed vector spaces). The condition number describes what happens to a problem when there is a small change in $x \in X$, i.e how much $f(x)$ changes for a small change in x. A well-conditioned problem implies a small change in $f(x)$ for a small perturbation in x, whereas a large change in $f(x)$ for the same would be called an ill-conditioned problem.

In floating point arithmetic, the errors introduced are relative in nature. So we look at the relative condition number of the problem. For the given problem f, its relative condition number (κ) can be given as:

$$\kappa = \sup_{\|\delta x\| \to 0} \left(\frac{\|\delta f\|}{\|f(x)\|} \right) \Big/ \left(\frac{\|\delta x\|}{\|x\|} \right) \tag{1.1}$$

The $\|.\|$ symbol here refers to the norm of the vector.

1.1.1 Condition Number of a Matrix

For linear systems $Ax = b$ ($A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ and $b \in \mathbb{R}^n$) given a matrix A, there can be three different ways the condition number can be interpreted.

Suppose for the given matrix A, input x is given and we are required to calculate b. In such a case, the change in output b corresponding to the perturbation in input x is examined [27]. Using equation 1.1, we can write the condition number of the problem as:

$$\kappa_F = \sup_{\|\delta x\|\to 0} \left(\frac{\|A(x+\delta x)) - Ax\|}{\|Ax\|}\right) \Big/ \left(\frac{\|\delta x\|}{\|x\|}\right)$$
$$\kappa_F = \sup_{\|\delta x\|\to 0} \left(\frac{\|A\delta x\|}{\|\delta x\|}\right) \Big/ \left(\frac{\|Ax\|}{\|x\|}\right)$$

Then using the definition of a matrix norm ($\sup_\delta \|A\delta x\| / \|\delta x\| = \|A\|$) and replacing Ax with b, we get:

$$\kappa_F = \frac{\|A\|\,\|x\|}{\|b\|} \tag{1.2}$$

We refer to this as the **forward condition number**. $\|.\|$ can be any type of norm.

Similarly, when the input is b and we are required to calculate x, the condition number for perturbations in b is given by:

$$\kappa_B = \sup_{\|\delta b\|\to 0} \left(\frac{\|A^{-1}(b+\delta b)) - A^{-1}b\|}{\|A^{-1}b\|}\right) \Big/ \left(\frac{\|\delta b\|}{\|b\|}\right)$$
$$\kappa_B = \sup_{\|\delta b\|\to 0} \left(\frac{\|A^{-1}\delta b\|}{\|\delta b\|}\right) \Big/ \left(\frac{\|A^{-1}b\|}{\|b\|}\right)$$

Then using the definition of a matrix norm ($\sup_{\delta b} \|A^{-1}\delta b\| / \|\delta b\| = \|A^{-1}\|$) and replacing $A^{-1}b$ with x, we get:

$$\kappa_B = \frac{\|A^{-1}\|\,\|b\|}{\|x\|} \tag{1.3}$$

We call κ_B as the **backward condition number**.

The above two quantities were obtained by perturbing x and b. We now perturb A infinites-

imally by δA and fix b. Then the change in x, δx must be such that:

$$(A + \delta A)(x + \delta x) = b$$
$$Ax + \delta Ax + A\delta x + \delta A\delta x = b$$

Using $Ax = b$ and ignoring $\delta A\delta x$, above expression becomes:

$$\delta Ax + A\delta x = 0$$
$$\delta x = -A^{-1}(\delta A)x \tag{1.4}$$

Because of the above equation, we can write the following:

$$\left(\frac{\|\delta x\|}{\|x\|}\right) \Big/ \left(\frac{\|\delta A\|}{\|A\|}\right) \leq \|A\| \left\|A^{-1}\right\|$$

When A is the perturbed quantity, supremum of the left hand side of equation 1.5 will be the condition number of that problem. Hence, we can say that:

$$\kappa = \|A\| \left\|A^{-1}\right\| \tag{1.5}$$

We call the above quantity as the **condition number of the matrix** A. It is visible upon little speculation that the condition number of a matrix is $\kappa = \kappa_B \times \kappa_F$. This quantity is an important indicator in many applications, as we will see later.

1.2 Iterative solvers

A primitive classification of Linear solvers would be into Direct and Iterative solvers. Direct solvers, though more robust, are cost prohibitive when size of matrix 'N' increases as they scale as $\mathcal{O}(N^3)$ [5]. The advantage of iterative schemes is that for each iteration, the operation count increases by a factor of $\mathcal{O}(N^2)$ only. In most cases they require only a few iterations (much less than N) to converge. Moreover, most of these methods are able to utilize the sparsity of the matrices very well. In fact for matrices with sizes in millions and billions, iterative methods are the only viable option.

We explore a certain subset of iterative methods called the Krylov subspace methods [14]. Specifically, we look at two Krylov Methods called the Conjugate Gradient Method and the GMRES (Generalized Minimal Residual) method.

1.2.1 Conjugate Gradient

The method of Conjugate Gradient can be interpreted as the solution to an optimization problem with the objective function $f(x) = x^T Ax - b^T x + c$ (where $A \in \mathbb{R}^{n \times n}$ is Symmetric positive definite [13], $b \in \mathbb{R}^n$, $c \in \mathbb{R}$ and $x \in \mathbb{R}^n$ is the parameter). For the objective function to have a maxima, the matrix A must be positive definite. Moreover, if the matrix A is symmetric, then solving the optimization problem is equivalent to solving $Ax = b$ because the function is convex ($\nabla^2 = A$ which is symmetric positive definite) and $\nabla f(x) = b - Ax$.

The method looks for the solution 'x' in A-conjugate directions. Two vectors 'p_i' and 'p_j' are said to be A-conjugate if $p_i^T A p_j = p_j^T A p_i = 0$. We choose x to be a linear combination of these search directions:

$$x = \sum_{j=1}^{n} \alpha_j p_j$$

For simplicity, we will assume the starting vector $x_0 = 0$ now and for our experiments too. A-conjugacy ensures linear independence of p_j. As a result, n such vectors spans $\mathbb{R}^n$.

Algorithm 1 Conjugate Gradient

Require: A, b

$x_0 = 0$
$r_0 = b - Ax_0$
$p_0 = r_0$
for $j = 1$ **to** N **do**
  $\alpha_j = \frac{r_{j-1}^T r_{j-1}}{p_{j-1}^T A p_{j-1}}$
  $x_j = x_{j-1} + \alpha_j p_{j-1}$
  $r_j = r_{j-1} - \alpha_j A p_{j-1}$
  $\beta_j = \frac{r_j^T r_j}{r_{j-1}^T r_{j-1}}$
  $p_j = r_j + \beta_j p_{j-1}$
end for

In algorithm 1, if the iteration is continued for 'N' steps, we converge to the exact x when the computing device has sufficient precision. Generally for iterative methods, we need the error/residue to be within a certain tolerance, and for a well conditioned matrix, the tolerance is reached much before N iterations. CG requires minimal additional storage space and is the method of choice for solving SPD systems.

1.3 Generalized Minimal Residual method(GMRES)

As the name suggests, the method aims to minimize the residue (specifically, the 2-norm of the residue) [25]. At the 'n^{th}' iteration, the method seeks the least squares solution of the n dimension Krylov subspace $K_n = K_n(A, b) = \text{span}\{b, Ab, A^2b, \ldots, A^{n-1}b\}$ [14]. GMRES calculates an orthonormal basis in each iteration using Arnoldi's iteration and finds the best y_n which minimizes the residue for the n^{th} iteration ($x_n = V_n y_n$ where V_n is the orthonormal basis for the n dimension Krylov subspace). Formally:

$$AV_n = V_{n+1}H_n$$

So the residue at the n^{th} iteration becomes:

$$\|Ax_n - b\| = \left\|H_n y_n - V_{n+1}^T b\right\|$$

In algorithm 2, we try to find the y_n that will minimize r_n. At the end of k iterations, $x_k = V_k y_k$. This poses a storage constraint on the method. For k iterations, all the k vectors must be stored. Hence, we generally use GMRES method with restarts [4].

GMRES was created as a generalized version of the Minimal Residual Method (MINRES) [21]. The MINRES method is for symmetric systems only whereas the GMRES methods applies to non-symmetric systems as well . Since we are limiting our experiments to SPD systems, we will also be in fact working with MINRES only. The only difference in the algorithm of MINRES is that instead of the Arnoldi iteration, we do the Lanczos iterations[12].

Algorithm 2 GMRES

Require: A, b, tol

$x_0 = 0$
$r_0 = b - Ax_0$
$v_1 = r_0 / \|r_0\|$
for $j = 1$ **to** N **do**
  $h_{i,j} =< Av_j, v_i >, i = 1, 2, ..., j$
  $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$
  $h_{j+1,j} = \|\hat{v}_{j+1}\|$
  $v_{j+1} = \hat{v}_{j+1} / h_{j+1,j}$
  if $relative\_residue < tol$ **then**
    break
  end if
end for
$x_k = x_0 + V_k y_k$

1.4 Preconditioning Linear Systems

Iterative schemes save operations and utilize the sparsity of the matrix to converge faster. But what they fail to provide is the robustness that Direct Solvers offer. At times the iterative scheme might not even converge.

Convergence properties of these linear systems have a great deal of dependence on the condition number of the matrix A. Systems with a higher condition number have been known to take more number of iterations to converge as compared to problems with a lower condition number [22]. Determining the condition number is in itself a very hard problem ($\mathcal{O}(n^3)$ iterations) and one can only get a rough estimate [8]. Preconditioners are applied to problems with the purpose of reducing the condition number, making the system easy to solve. There are three general ways of applying a preconditioner to a linear system $Ax = b$:

- Left preconditioning : $M^{-1}Ax = M^{-1}b$.

  The matrix M is similar to the matrix 'A'. Hence $M^{-1}A$ should resemble an identity matrix which has a condition number of 1.

- Right preconditioning: $AM^{-1}Mx = b$

  In this case, we solve the system $AM^{-1}y = b$ first and then we solve for x, since $x = M^{-1}y$. Same as the case above, a matrix M which is similar to A will speed up convergence.

- Split preconditioning: $M_1^{-1}AM_2^{-1}M_2x = M_1^{-1}b$

  The matrix M splits into M_1 and M_2 ($M = M_1M_2$). The system $M_1^{-1}AM_2^{-1}y = M_1^{-1}b$ is then solved. Following that, x is solved like so: $x = M_2^{-1}y$

The way to apply preconditioners is limited to these three but techniques like adaptive and variable preconditioning might be also used to get a better output [18, 3]. For a non-singular M, we can see that the solution of the problem does not change, but the problem itself changes.

Fig 1.1 shows the effect a preconditioner can have on the convergence properties of a linear system. As evident from the figure, the advantage of using a preconditioner is quite significant. The residue after 40 iterations is around 10^{-7} in the preconditioned case whereas in the non-preconditioned case, it managed to reach 10^{-1} only.

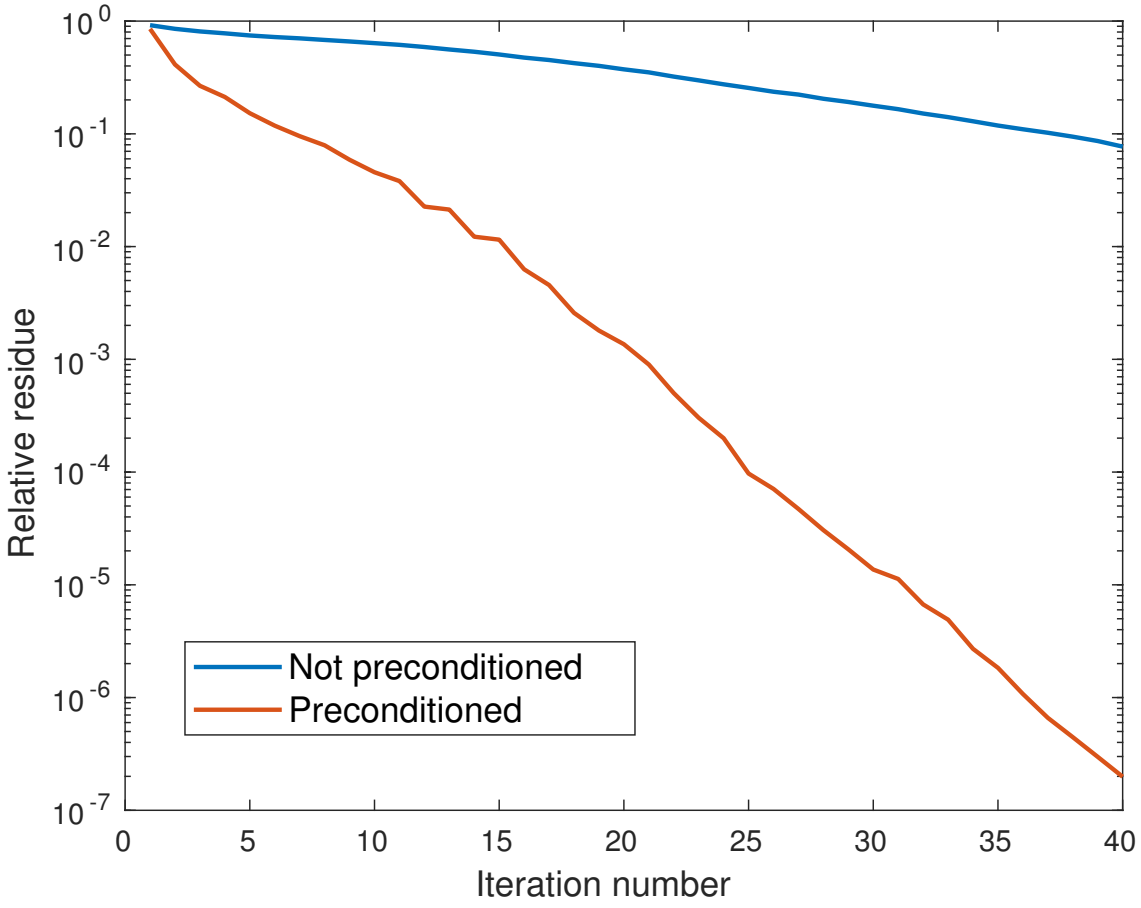

Figure 1.1: The above figure shows comparison of convergence characteristics of a matrix solved using Conjugate Gradient with and without a preconditioner

1.4.1 Incomplete Cholesky Preconditioner

The Cholesky factorization decomposes a Symmetric Positive Definite Matrix A into the product of a lower triangular matrix L and its transpose.

$$A = LL^T$$

This method completely factorizes the matrix, but when the matrix is large and sparse, we can seek for a sparse approximation of L too. The level of fill-in decides the sparsity pattern of this sparse approximation $\tilde{L}$. A very common practice is to choose sparsity pattern of L similar to that of A. This incomplete factorization of A is used as a preconditioner and called the Incomplete Cholesky preconditioner (zero fill-in)[17]. Higher the level of fill-in [11], more the number of extra operations required. For large matrices, a high level of fill-in can be a huge overhead. Hence generally for large and sparse systems, zero fill-in is the default choice. If the incomplete factorization gives us $\tilde{L}$, then :

$$A = \tilde{L}\tilde{L}^T + E, \tilde{L}(i,j) = 0 \text{ if } (i,j) \notin S$$

where E is the error matrix and S is the sparsity pattern of A. Generally increasing the fill-in leads to a smaller E. For complete factorization, E will be a zero matrix. For this incomplete factorization to succeed, it is not sufficient for the matrix A to be symmetric positive definite (SPD). The '$A - E$' matrix must be SPD ($A - E = \tilde{L}\tilde{L}^T$) too. This is not always true which results in the factorization itself producing negative pivots. There exist ways to make sure that the factorization succeeds, such as using diagonal shifts, recasting the factorization, using A-orthogonalization and so on [6, 16].

The advantage of the preconditioner is that it conserves symmetry and positive definiteness when applied to an SPD system. The Incomplete Cholesky factorizations results in the lower triangular matrix $\tilde{L}$ and let $\hat{L}$ ($= \tilde{L}^{-1}$) be its inverse, then the preconditioner is applied in the following way :

$$\hat{L}A\hat{L}^T\hat{L}x = \hat{L}b$$

1.4.2 Split Jacobi preconditioner

The Jacobi preconditioner is essentially the inverse of the diagonal matrix of A. This is applied as a left or right preconditioner to the matrix A [19]. So the Jacobi preconditioner 'D' is simply:

$$D = diag(A)$$

This preconditioner is especially useful for diagonally dominant matrices. But for methods like Conjugate Gradient, this preconditioner does not ensure an SPD system when applied. So we apply a modified version of the Jacobi preconditioner, called the split Jacobi preconditioner.

Let $\tilde{D}$ be the split Jacobi preconditioner, then:

$$\tilde{D}^2 = diag(A)$$

When A is Symmetric and Positive Definite, its diagonal contains only positive values, ensuring the existence of $\tilde{D}$ always.

We call this the split Jacobi Preconditioner which breaks the diagonal matrix into two by taking its square root. This preconditioner is applied in a split way, the same as we do in incomplete cholesky preconditioner, i.e. for $\hat{D} = \tilde{D}^{-1}$:

$$\hat{D}A\hat{D}\hat{D}x = \hat{D}b$$

Above is the resultant system we end up after applying the Split Jacobi Preconditioner.

1.5 Related Work

1.5.1 Jacobi vs. ILU preconditioner

There are a rich source of methods for solving linear equations nowadays such as BiCG, QMR, IDR, CGS. These methods are generally accompanied by a preconditioner which increases the operation cost but supposedly reduces the iteration count by much more. In their paper [2] Anzt et al. look at two such preconditioners i.e. the ILU and Jacobi preconditioners.

The ILU (Incomplete LU) preconditioner is similar to the Incomplete Cholesky preconditioner in the sense that it also factorizes the matrix incompletely based on the level of fill-in. For zero fill-in, the sparsity pattern of L matches to that of the lower triangular part of A and sparsity pattern of U matches the upper triangular part of A.

In his paper Anzt applies the same ILU and Jacobi preconditioners indiscriminately to the set of matrices acquired from the Florida Sparse Matrix collection. Their purpose was to determine if the increased number of operations per iteration was justified by the reduced number of iterations or not. They compared the speed-up (in terms of time taken) obtained by the application of these preconditioners and it was found that even though ILU was a better preconditioner (in terms of iterations reduced), it failed to provide the speedup that Jacobi gave because of the increased number of operations ILU took. Moreover their results indicated that Jacobi was a more reliable preconditioner. ILU preconditioners managed to increase the

time taken to converge in more than half the cases whereas Jacobi gave a speed-up more than one in most cases. This introduces the need for robustness of preconditioners as discussed in the next section.

1.5.2 Robustness of preconditioners

As discussed in the previous section, randomly applied preconditioners may not give the desired result. Chow [7] explains the need for robust preconditioning techniques and explains the pitfalls that come with using preconditioners like ILU. For instance ILU(0) [10] produces very ill-conditioned L and U for a non-symmetric system [24]. Even techniques like thresholding and pivoting cannot ensure that the preconditioning will succeed. There have been continued efforts still to stabilize the ILU preconditioner [1].

Further Benzi [6] in his paper does an in-depth survey for existing preconditioning techniques, their existence, stability and again stresses on the need for robustness of these preconditioners. Zhang and Saad suggest such a multipurpose preconditioner in their paper [28]. We put our focus on preconditioners for the SPD systems and study their properties as affected by factors like condition number, tolerance and size. A benchmark for the gain obtained in terms of iterations can help developers and engineers decide the need for a preconditioner.

1.5.3 Error Estimator for preconditioned systems

Puneet [15] discusses in his paper the significance of error-estimators for methods like Bi-CG and GMRES. High condition number matrices tend to bring with them a high magnitude of uncertainty. For example, a matrix with condition number κ, the relative error at k^{th} iteration ($\epsilon_k = x^* - x_k$) can be bounded by the relative residue at the k^{th} iteration ($r_k = b - Ax_k$) in the following way:

$$\frac{\|r_k\|}{\|b\|\,\kappa} \leq \frac{\|\epsilon_k\|}{\|x^*\|} \leq \frac{\|r_k\|\,\kappa}{\|b\|}$$

Resultantly, in cases when the condition number is too high, an error estimator might be necessary to avoid stopping prematurely or too late. This paper [15] discusses the accuracy of residue as an estimate of the error and also how this "uncertainty" is affected by factors like condition number of the matrix, forward condition number and backward condition number of the problem. They show that for high condition number problems, the residue can be many orders of magnitude larger or smaller the actual error, especially for problems like GMRES. GMRES specifically minimizes the residue, so one can expect the residue to be much lower value than the actual error. And as the condition number of the problem increases, so does the difference between the residue and error. Preconditioning aims to reduce the condition number

of the matrix. So we explore the significance of error estimators for these preconditioned systems.

1.6 Objectives

We saw how preconditioners can drastically improve the convergence properties of the system. It was also noted that applying these preconditioners indiscriminately might not have the desired effect [2]. Hence we try to quantify the computing saved.

We investigate solvers like CG, where we care about preserving symmetry. Only split preconditioners do so for SPD matrices:

- The Incomplete Cholesky Factorization
- The Split Jacobi preconditioner

Another advantage of using SPD matrices is that they make the theoretical bounds (discussed in the next chapter) tighter. We try to quantify the iterations saved by these preconditioner which helps us establish their robustness [6].

Although CG is the preferred solver for SPD matrices, it converges very slowly for high conditioned matrices. GMRES on the other hand converges much faster in such cases. So we include GMRES in our analysis too. Later for the same set of solvers and preconditioners, we extend the work done by my peers [15] and demonstrate the significance of error-estimators.

Chapter 2

Theoretical estimate of gains due to preconditioners

In this chapter we try to derive an estimate for the iteration gain provided by preconditioners. We start by simplifying the Chebyshev polynomial which will be further used to simplify the upper bounds obtained in GMRES and CG. This helps us bound the iterations taken and ultimately arrive to our estimate.

2.1 Chebyshev Polynomial

Let $w = t + \sqrt{t^2 - 1}$ and $|t| > 1$, then the i^{th} degree Chebyshev polynomial is given by:

$$C_i(t) = \frac{1}{2}(w^i + w^{-i}) \tag{2.1}$$

Also, the i^{th} Chebyshev polynomial can be either an odd or even function, depending on the value of i. In general we can say that $|C_i(t)| = C_i(|t|)$ and resultantly:

$$\left| C_i\left(\frac{1+c}{1-c}\right)\right| = C_i\left(\frac{c+1}{c-1}\right) \tag{2.2}$$

Putting $t = (c+1)/(c-1)$, $w = (\sqrt{c}+1)/(\sqrt{c}-1)$. Thus for a large i, $w^i + w^{-i} \approx w^i$ and $w^i + w^{-i} \geq w^i$. Combining this with the equation 2.1 and 2.2 we get:

$$\frac{1}{|C_i(\frac{1+c}{1-c})|} \leq 2\left(\frac{\sqrt{c}-1}{\sqrt{c}+1}\right)^i \tag{2.3}$$

2.2 Upper Bound for Relative Residue (GMRES)

We follow the analysis done by Saad [23]. GMRES minimises the residue $(b - Ax)$ obtained from the Krylov subspace at any step k for a positive definite matrix A. Hence ($\|.\|$ signifies the L2 norm henceforth):

$$\begin{aligned} \|r_k\| &= \min_{p \in \mathbb{P}^k} \|p(A) r_0\| \\ \implies \|r_k\| &\leq \min_{p \in \mathbb{P}^k} \|p(A)\| \, \|r_0\| \\ \implies \frac{\|r_k\|}{\|r_0\|} &\leq \min_{p \in \mathbb{P}^k} \|p(A)\| \end{aligned} \tag{2.4}$$

$\mathbb{P}^k$ is the set of all monic polynomials ($p(0) = 1$) with degree less than or equal to k, and r_k refers to the residue after k^{th} iteration.

For all our experiments, we start with a zero vector x_0, meaning $r_0 = b$. Assuming that A can be diagonalized to $A = V \Lambda V^{-1}$, equation 2.4 becomes:

$$\begin{aligned} \frac{\|r_k\|}{\|b\|} &\leq \min_{p \in \mathbb{P}^k} \left\| V p(\Lambda) V^{-1} \right\| \\ \implies \frac{\|r_k\|}{\|b\|} &\leq \min_{p \in \mathbb{P}^k} \|V\| \, \|p(\Lambda)\| \, \left\| V^{-1} \right\| \end{aligned} \tag{2.5}$$

Now, $p(\Lambda)$ is simply a diagonal matrix with the i^{th} diagonal entry being $p(\lambda_i)$, where λ_i is the i^{th} eigenvalue. Therefore :

$$\|p(\Lambda)\| = \max_{\lambda \in \Lambda} |p(\lambda)|$$

So, we can re-write equation 2.5 as:

$$\frac{\|r_k\|}{\|b\|} \leq \kappa(V) \min_{p \in \mathbb{P}^k} \max_{\lambda \in \Lambda} |p(\lambda)| \tag{2.6}$$

Here $\kappa(.)$ refers to the condition number of the matrix. Also, when λ is real (α, β are the smallest and the largest eigenvalues):

$$\min_{p \in \mathbb{P}^k} \max_{\lambda \in \Lambda} |p(\lambda)| \leq \frac{1}{|C_k(\frac{\alpha+\beta}{\alpha-\beta})|} \tag{2.7}$$

Where 'C_k' is the Chebyshev polynomial of degree 'k'. Let 'κ' be the condition number of an

SPD system A. Then, after simplifying the Chebyshev polynomial as seen in equation 2.3, we can rewrite equation 2.6 as:

$$\frac{\|r_k\|}{\|b\|} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \tag{2.8}$$

To check how close the upper bound is to the actual quantity, we generate random SPD matrices of size $N = 500$. We run GMRES for a randomly generated non-trivial right hand side b to find an x that solves $Ax = b$. Then the ratio of the upper-bound and observed relative residue is averaged over all iterations for these sample matrices. Figure 2.1 has the ratio plotted for 1000 such sample matrices. We see that the ratio averages to 6.83 over all such matrices. This means the upper bound can be used for analysis.

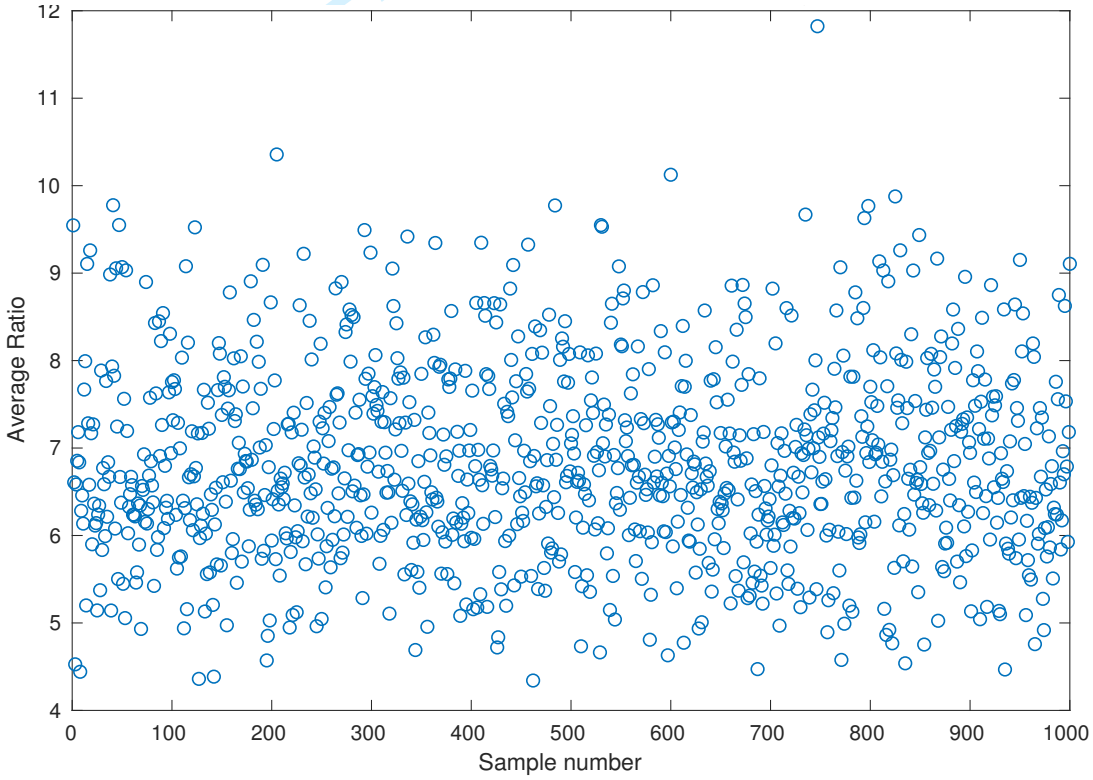

Figure 2.1: Ratio of upper bound and observed relative residue ($\kappa = 10^3, N = 500$) avg = 6.83 . This establishes the utility of the theoretical upper bound derived.

2.3 Upper Bound for Relative A-norm (CG)

A-norm of the error is given by

$$\|\epsilon_k\|_A = \sqrt{(x^* - x_k)^T A (x^* - x_k)}$$

x_k being the k^{th} iterate of x and x^* being the true solution for $Ax = b$. Similar to GMRES, Saad's analysis for CG [23] shows that:

$$\frac{\|\epsilon_k\|_A}{\|x^*\|_A} \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k$$

Similar to what we did for GMRES, here also we generate 1000 random SPD matrices. They are solved using the Conjugate Gradient method. Ratios of the upper-bound to the actual quantity are calculated over iterations and averaged. The ratio is plotted for 1000 such matrices as shown in Figure 2.2.

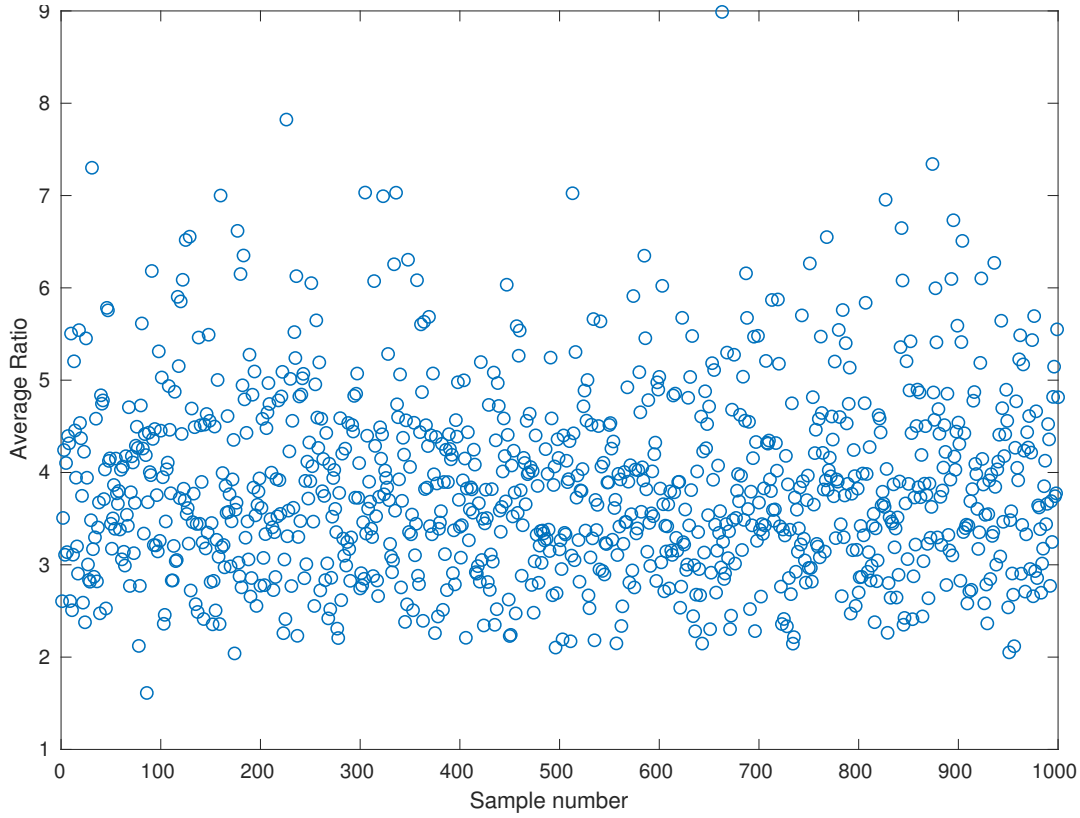

Figure 2.2: Ratio of upper bound and observed relative A-norm ($\kappa = 10^3, N = 500$) avg = 3.85 . This establishes the utility of the theoretical upper bound derived.

2.4 Bounds on the gain

Suppose we want to stop the iterations when relative residue $< \epsilon$ and at iteration 'i' we achieve a tolerance less than ϵ. Then using equation 2.8:

$$\begin{aligned} \frac{\|r_i\|}{\|b\|} &\leq \epsilon \\ \implies 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i &\leq \epsilon \\ \implies i \log\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right) &\geq \log\left(\frac{2}{\epsilon}\right) \\ \implies i &\geq \frac{\log\left(\frac{2}{\epsilon}\right)}{\log\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)} \end{aligned} \tag{2.9}$$

What the above implies is that when we need to reach a tolerance that is less than ϵ, then we need to run the algorithm for at most as many times as the ceiling of the quantity on the left hand side of Equation 2.9.

To verify the validity of Equation 2.9, we fix the condition number and the distribution of singular values after which we generate random matrices of different sizes. For the right hand side b, we use the sum of the singular vectors of A. Then these system are solved using the GMRES method. Figures 2.3 to 2.6 are plots for condition numbers 10^2 to 10^5 respectively. The y axis shows the number of iterations taken to converge to 10^{-5} and x-axis shows the size of matrix. What we see is that as these matrices grow in size, they seem to become asymptotic to a certain line. This illustrates Equation 2.9 and the dependence on condition number and size. Higher condition numbers appear to draw the iterations taken, away from the upper bound, whereas increasing size brings it closer to the upper bound. This implies that for low condition numbers and higher sizes, we operate very close to the upper bound.

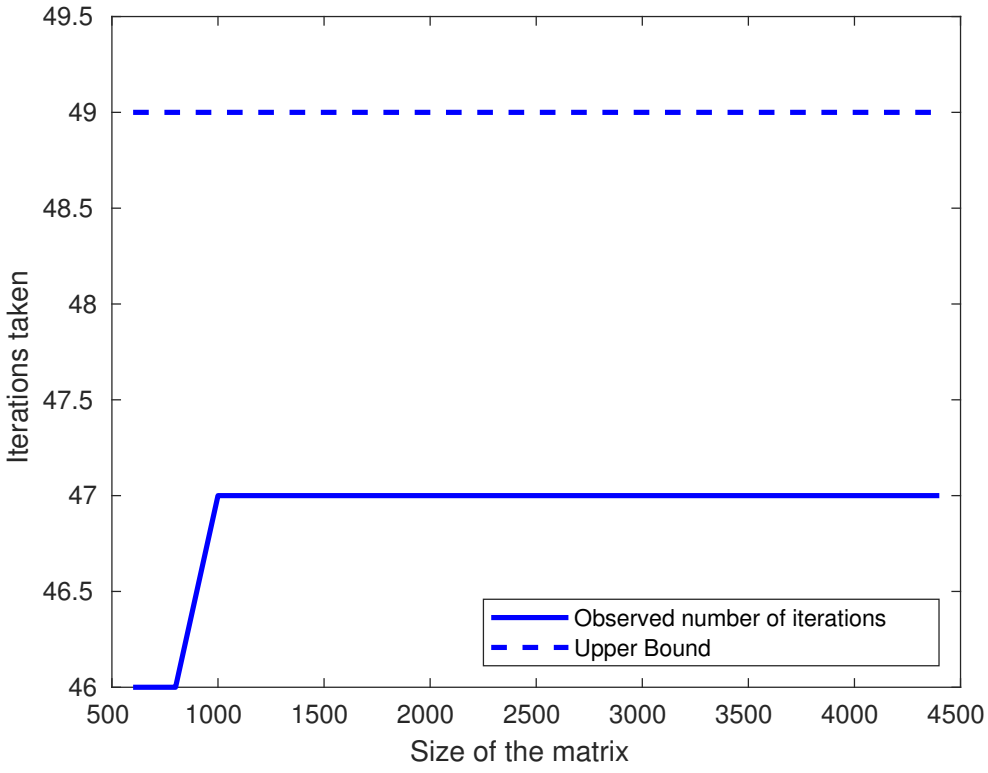

Figure 2.3: Comparison of the iterations taken to the upper bound for a condition number of 10^2

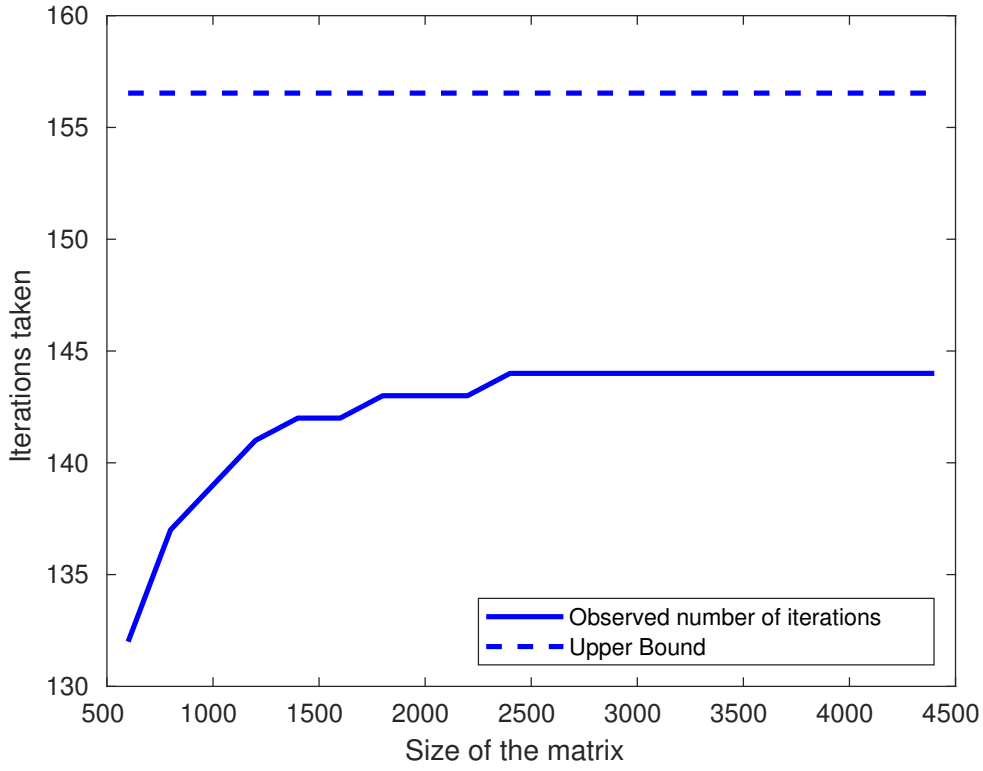

Figure 2.4: Comparison of the iterations taken to the upper bound for a condition number of 10^3

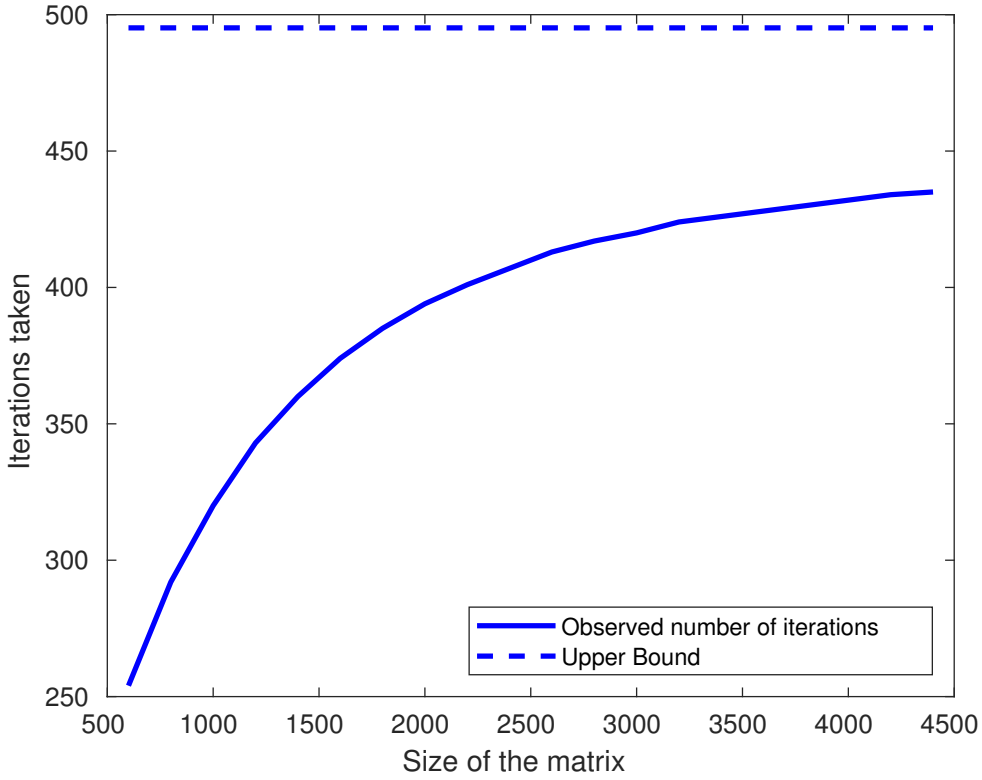

Figure 2.5: Comparison of the iterations taken to the upper bound for a condition number of 10^4

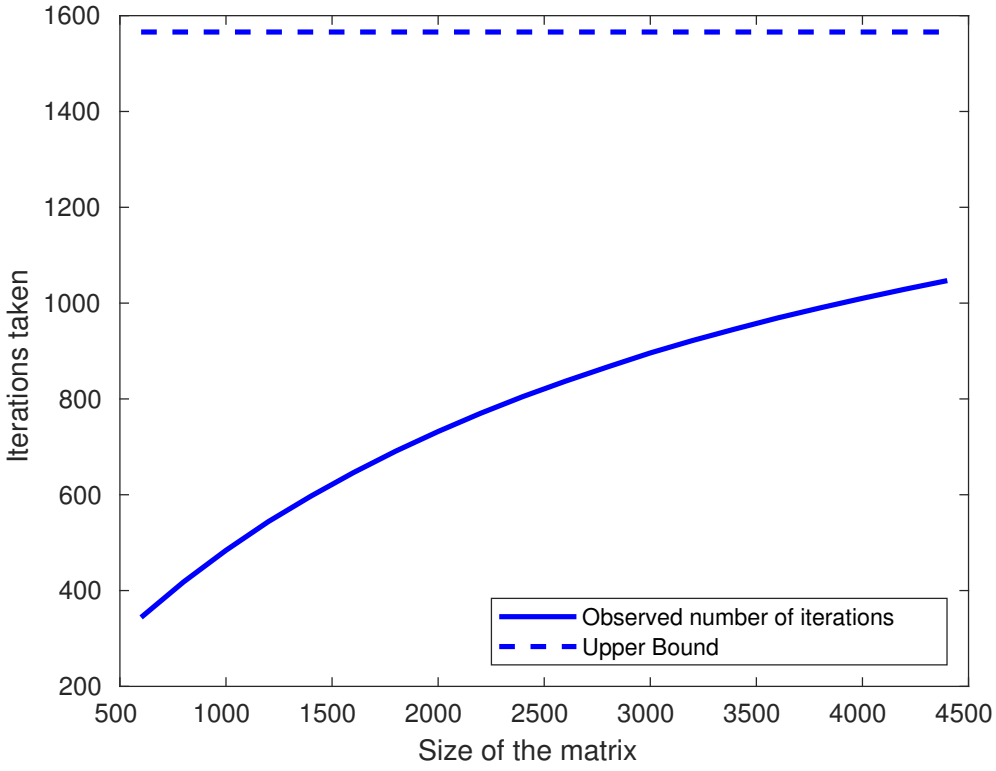

Figure 2.6: Comparison of the iterations taken to the upper bound for a condition number of 10^5

2.5 Gain

Assuming the left side 'b' is never trivial, and from equation (2.9), we get the below bounds on the number of iterations:

$$1 \leq i \leq \frac{\log\left(\frac{2}{\epsilon}\right)}{\log\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)}$$

For a preconditioned matrix, lets call its condition number as κ_p. For the preconditioned matrix too, the below is true:

$$1 \leq i_p \leq \frac{\log\left(\frac{2}{\epsilon}\right)}{\log\left(\frac{\sqrt{\kappa_p}+1}{\sqrt{\kappa_p}-1}\right)}$$

From the above two equations, one can bound the iterations gain (i/i_p) by:

$$\frac{\log\left(\frac{\sqrt{\kappa_p}+1}{\sqrt{\kappa_p}-1}\right)}{\log\left(\frac{2}{\epsilon}\right)} \leq \frac{i}{i_p} \leq \frac{\log\left(\frac{2}{\epsilon}\right)}{\log\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)} \tag{2.10}$$

2.6 Proposed estimate for the gain

From equation (2.10), we get the lower and upper bound of the gain. The square of the geometric mean of these bounds is :

$$G = \frac{\log\left(\frac{\sqrt{\kappa_p}+1}{\sqrt{\kappa_p}-1}\right)}{\log\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)} \tag{2.11}$$

Above is our estimate of the iteration gain for preconditioned CG and GMRES. It is important to note that the above may not be suited for small N.

Chapter 3

Numerical results

3.1 Experimental Setup

3.1.1 Matrices

Matrices of size 1000×1000 were generated using the *sprandsym* method in MATLAB. This *sprandsym* function generates symmetric sparse matrices of the specified density and condition number. We decided on a matrix density of $\mathcal{O}(NlogN)$ (Here N=1000, dimension of the matrix).

Alternatively, we also used few sparse matrices used in practical applications from SuiteSparse Matrix collection (Formerly the University of Florida Sparse Matrix Collection) [9]. The results of these matrices have been mentioned separately in table 3.4.

3.1.2 Preconditioners

The above matrices have two different kind of preconditioners applied to them: Split Jacobi and Incomplete Cholesky ($A \approx LL^T$). The Incomplete Cholesky has zero fill in, meaning the resultant matrix L will have a sparsity pattern same as that of the original matrix A.

3.1.3 Solvers

We use the GMRES and Conjugate Gradient Method for solving the system themselves and also their preconditioned counter-parts. As mentioned earlier, the preconditioners are applied such that they retain their symmetric positive definite property. Hence conditions for convergence are met for GMRES and Conjugate Gradient for the preconditioned matrices too.

For the purposes of analysis, we use the GMRES method without restarts.

3.1.4 Stopping criteria

We use the relative residue of the matrix to stop at a specific tolerance for GMRES as well as Conjugate Gradient. We use the same quantity $\|b - Ax\| / \|b\|$ for the preconditioned case as a stopping criteria. Even though for a preconditioned system, the residue would have a different value and meaning, we chose to keep the quantity similar for both of them because we are ultimately interested in solving $Ax = b$.

3.1.5 Gain Comparison

In our experiments below, the gain is calculated as the ratio of the number of iterations taken to converge without a preconditioner to the number of iterations taken with a preconditioner. The results are then compared with our estimate (Theoretical Gain) as depicted in equation 2.11.

3.2 Plots (Split Jacobi — Incomplete Cholesky)

For the plots below, we will see the gain for the Split Jacobi (left) and the Incomplete Cholesky preconditioner (right). For each Ax=b problem solved, the observed gain for GMRES, Conjugate Gradient is plotted along with our theoretical estimate.

3.2.1 Constant condition number, Varying tolerances

The condition number for the matrix is kept constant at 100 and the tolerances are varied from 10^{-3} to 10^{-6}. Figures 3.5 to 3.8 show the plots for the same. For better readability of the scatter plots, one can refer to table 3.1 for the mean values of the gain.

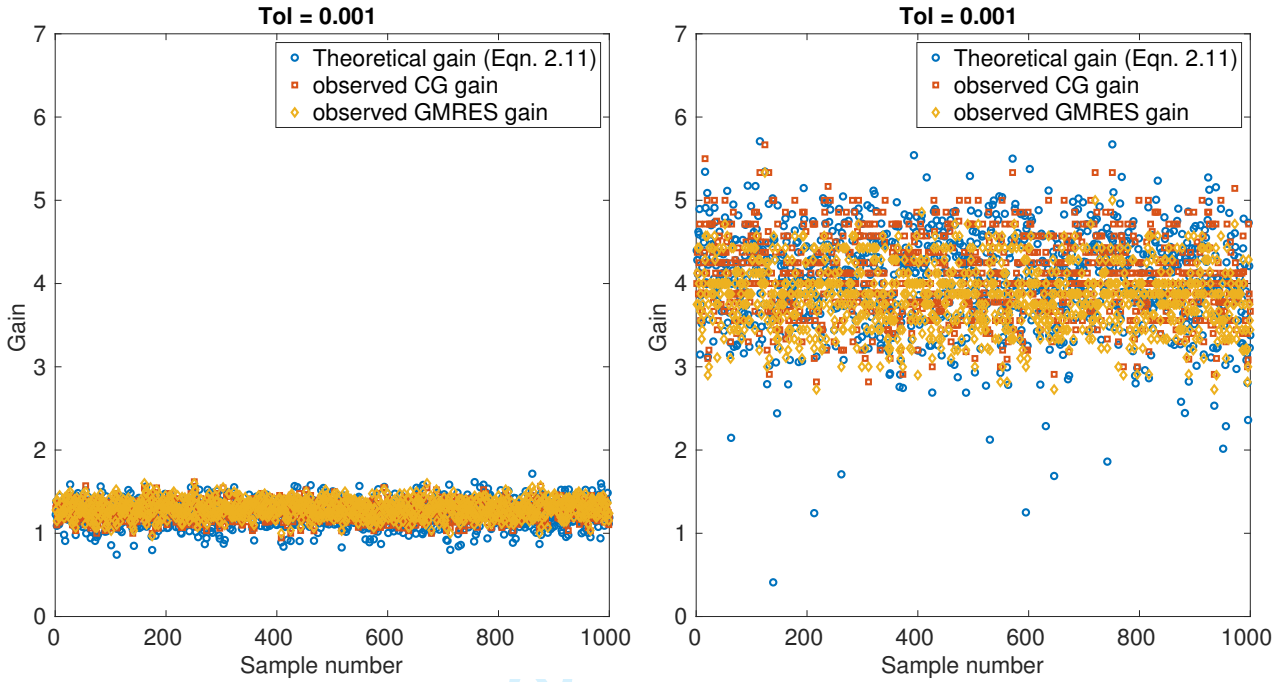

Figure 3.1: Observed and Theoretical gain comparison for Split Jacobi (left) and Incomplete Cholesky (right) preconditioner, for a matrix of size 1000×1000 and condition number 100 with a stopping tolerance of 10^{-3}

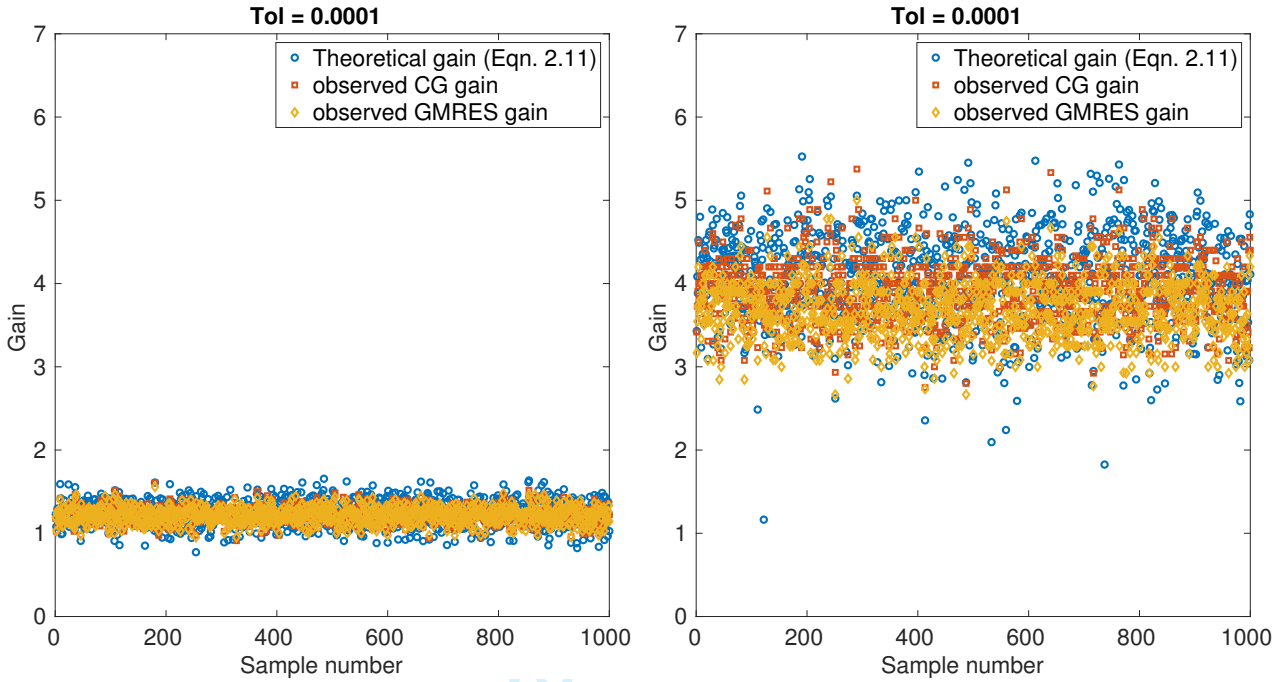

Figure 3.2: Observed and Theoretical gain comparison for Split Jacobi (left) and Incomplete Cholesky (right) preconditioner, for a matrix of size 1000×1000 and condition number 100 with a stopping tolerance of 10^{-4}

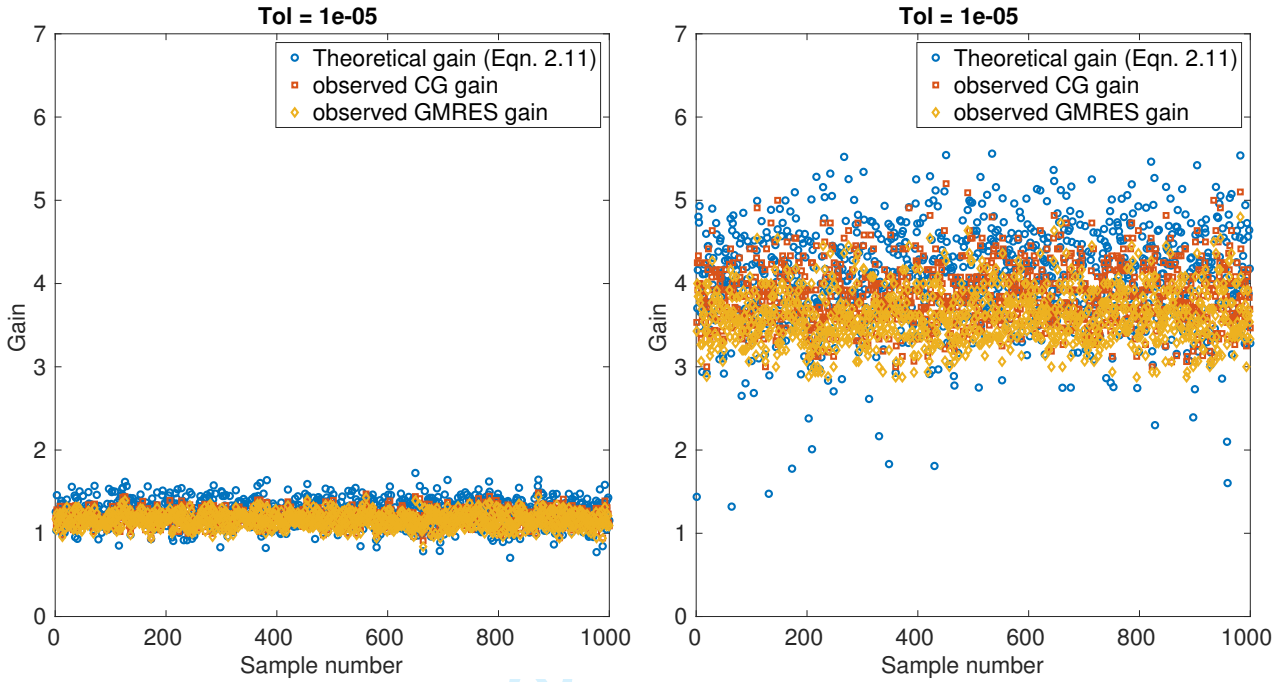

Figure 3.3: Observed and Theoretical gain comparison for Split Jacobi (left) and Incomplete Cholesky (right) preconditioner, for a matrix of size 1000×1000 and condition number 100 with a stopping tolerance of 10^{-5}

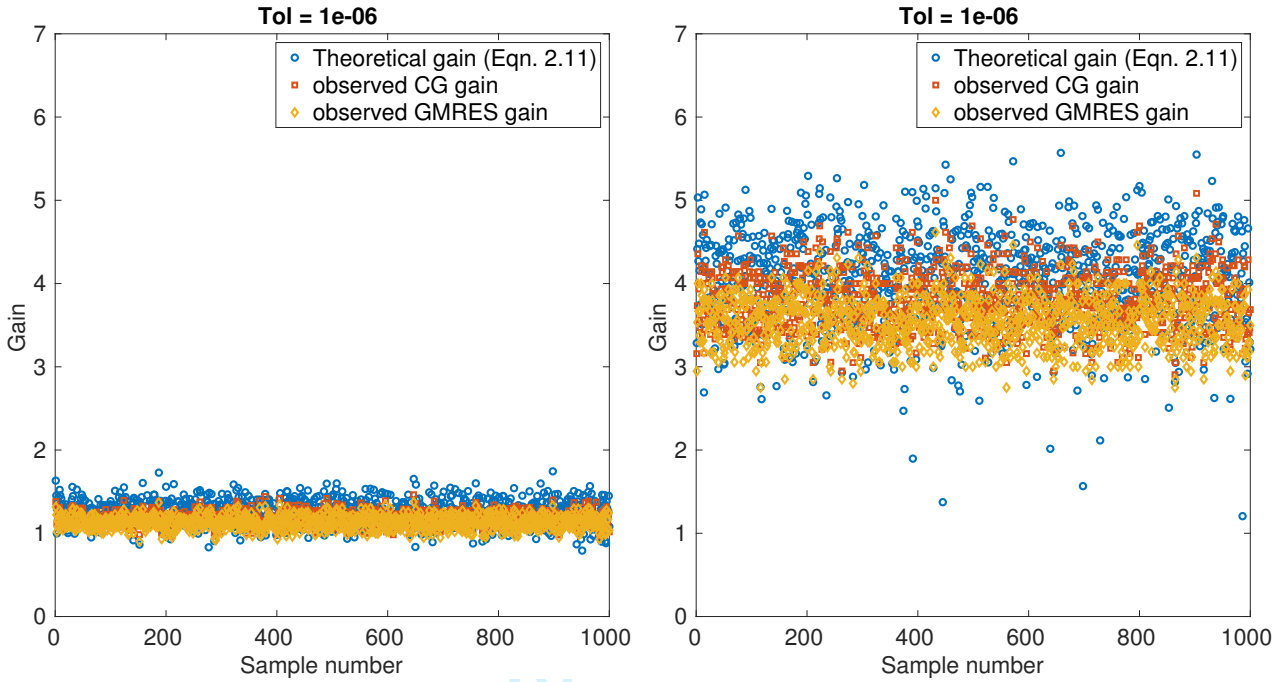

Figure 3.4: Observed and Theoretical gain comparison for Split Jacobi (left) and Incomplete Cholesky (right) preconditioner, for a matrix of size 1000×1000 and condition number 100 with a stopping tolerance of 10^{-6}

	Split Jacobi mean Gains			Incomplete Cholesky mean Gains		
Tolerance	Theoretical Gain	CG Gain	GMRES gain	Theoretical Gain	CG Gain	GMRES gain
10^{-3}	1.24	1.26	1.30	4.05	4.08	3.84
10^{-4}	1.24	1.21	1.21	4.08	3.92	3.69
10^{-5}	1.24	1.18	1.16	4.05	3.85	3.61
10^{-6}	1.25	1.17	1.13	4.06	3.81	3.55

Table 3.1: Mean of the respective gains for randomly generated matrices of condition number 100

3.2.2 Varying condition number, Constant tolerance

Now we keep the tolerance fixed at 10^{-5} and vary the condition number from 10^2 to 10^5 as seen in figures 3.5 to 3.8. For better readability of the scatter plots, one can refer to table 3.2 for the mean values of the gain.

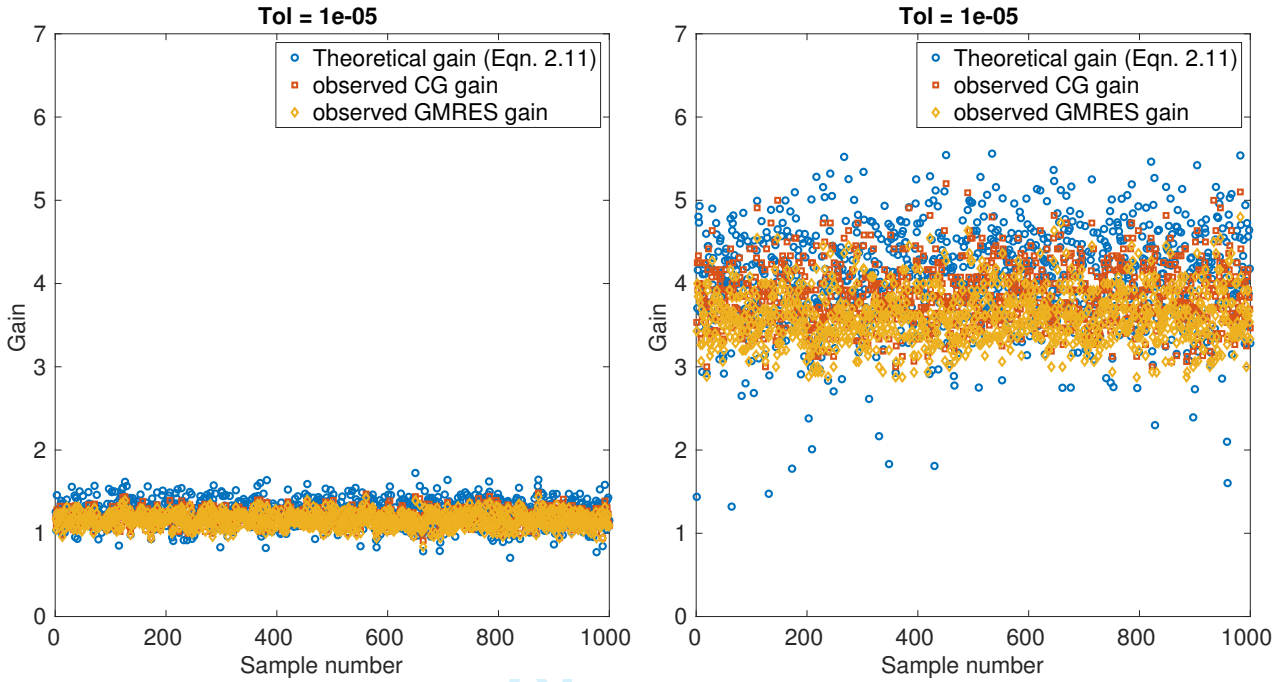

Figure 3.5: Observed and Theoretical gain comparison for Split Jacobi (left) and Incomplete Cholesky (right) preconditioner, for a matrix of size 1000×1000 and condition number 10^2 with a stopping tolerance of 10^{-5}

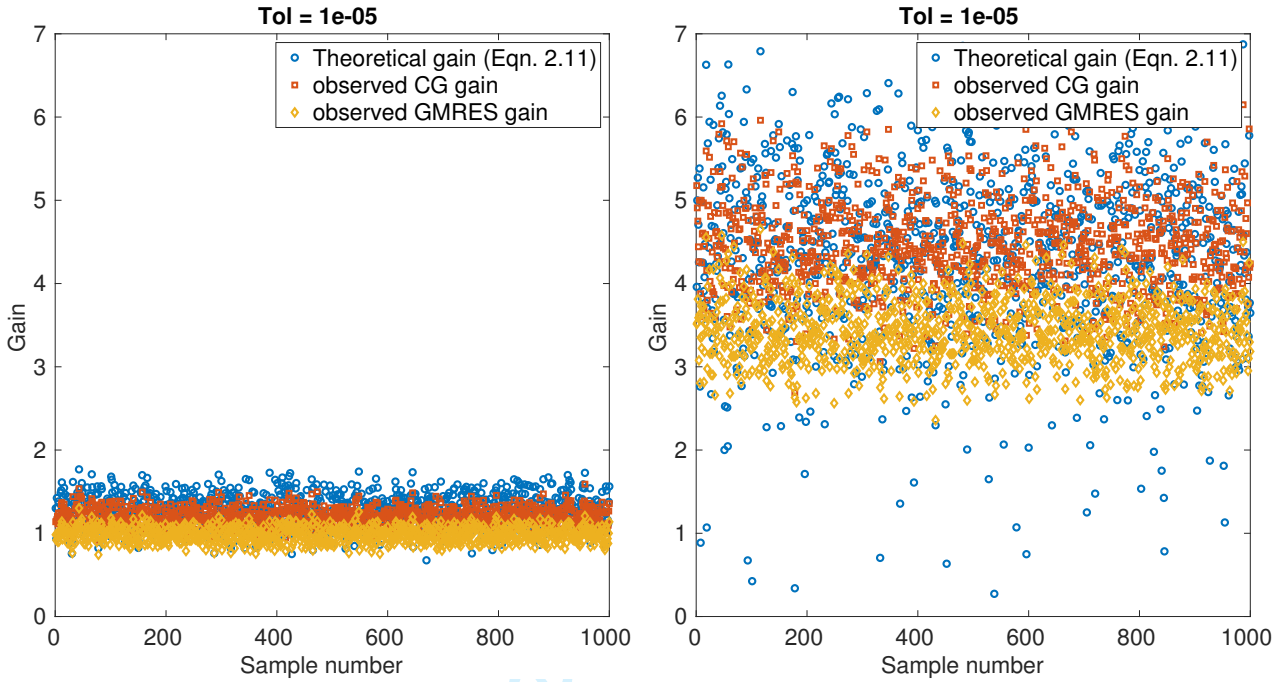

Figure 3.6: Observed and Theoretical gain comparison for Split Jacobi (left) and Incomplete Cholesky (right) preconditioner, for a matrix of size 1000×1000 and condition number 10^3 with a stopping tolerance of 10^{-5}

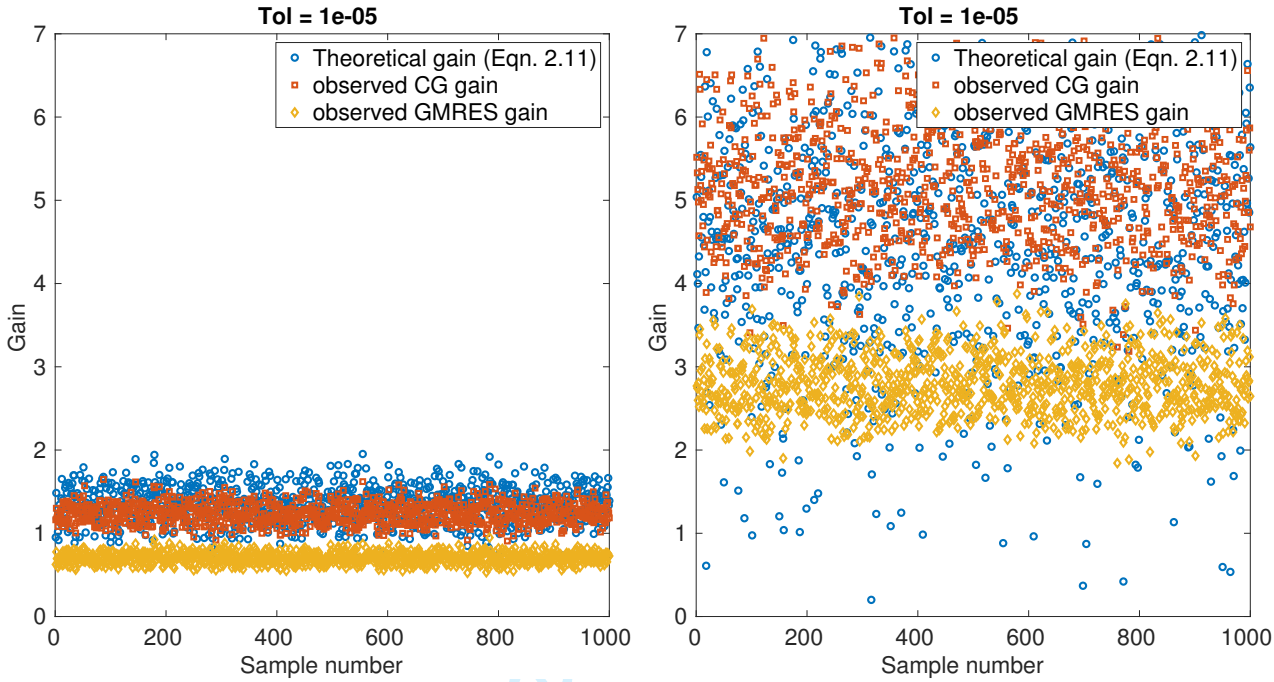

Figure 3.7: Observed and Theoretical gain comparison for Split Jacobi (left) and Incomplete Cholesky (right) preconditioner, for a matrix of size 1000×1000 and condition number 10^4 with a stopping tolerance of 10^{-5}

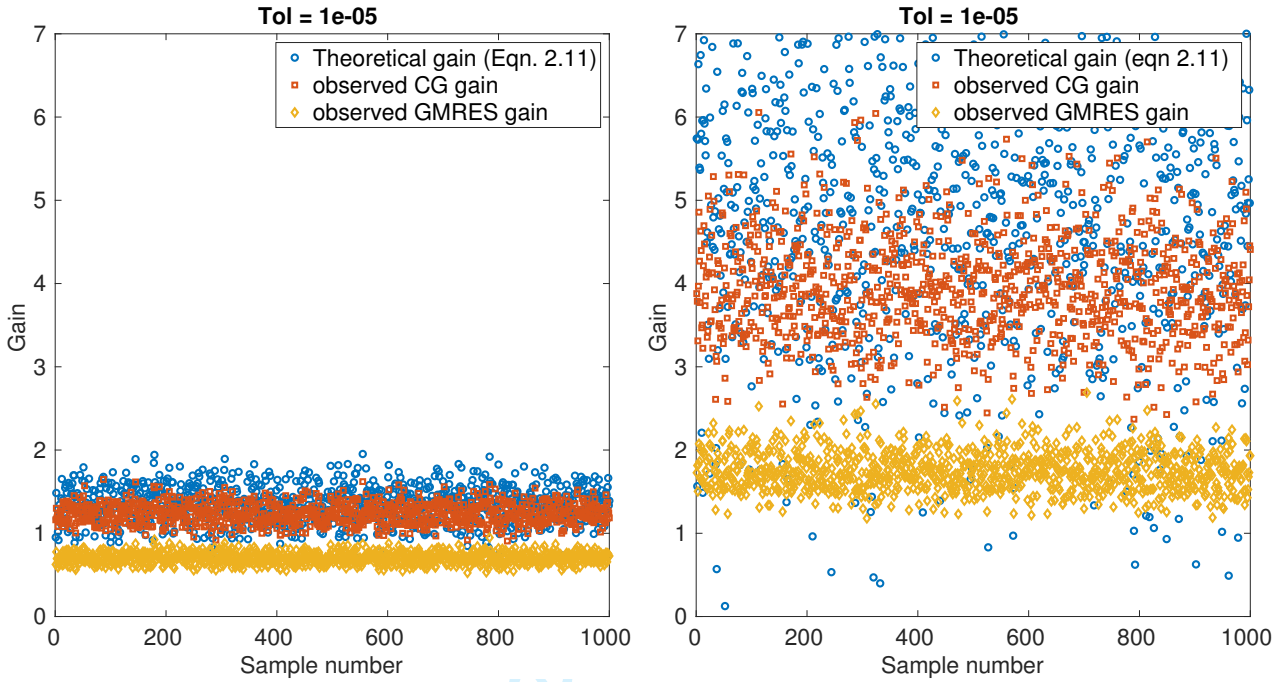

Figure 3.8: Observed and Theoretical gain comparison for Split Jacobi (left) and Incomplete Cholesky (right) preconditioner, for a matrix of size 1000×1000 and condition number 10^5 with a stopping tolerance of 10^{-5}

	Split Jacobi mean Gains			Incomplete Cholesky mean Gains		
Condition Number	Theoretical Gain	CG Gain	GMRES gain	Theoretical Gain	CG Gain	GMRES gain
10^2	1.25	1.17	1.13	4.06	3.81	3.55
10^3	1.28	1.20	0.95	4.36	4.52	3.40
10^4	1.33	1.23	0.67	4.70	5.26	2.68
10^5	1.38	1.00	0.58	5.13	3.46	1.66

Table 3.2: Mean of the respective gains for randomly generated matrices with a stopping tolerance of 10^{-5}

3.3 Results for the practical application matrices

As mentioned earlier, we also ran our experiments on some matrices that we find in real world use. These matrices are mostly high in condition number and hence the results for lower condition numbers are not available in this case. For these we solved for 100 random b for each matrix and have tabulated the gains obtained in table 3.4. The tolerance was set as 10^{-5} for these matrices. For simplicity, we will be referring to these matrices by the number assigned to them (Matrix no. 1, Matrix no. 2 and so on).

No.	Matrix Description	Type of problem	Size(N)	Link
1	S Admittance Matrix	Power Network Problem	1138	link
2	Stiffness Matrix	Structural Problem	1074	link
3	FE Approximation to biharmonic on plate	Structural Problem	960	link
4	S Admittance Matrix	Power Network Problem	685	link
5	Matric from MSC/NASTRAN	Structural Problem	726	link
6	Mixed-effects model	Statistical/Mathematical Problem	2541	link
7	Free Vibration Mass Matrix	Materials Problem	4875	link
8	S Admittance Matrix	Power Network Problem	494	link
9	S Admittance Matrix	Power Network Problem	662	link
10	Structure from NASA	Structural Problem	2416	link

Table 3.3: Matrix Descriptions

No.	Original Condition no.	Split Jacobi mean Gains			Incomplete Cholesky mean Gains		
		Theoretical Gain	CG Gain	GMRES gain	Theoretical Gain	CG Gain	GMRES gain
1	$10^{6.93}$	4.18	1.18	0.60	20.59	8.72	3.89
2	$10^{7.41}$	83.01	6.99	5.07	562.36	39.50	18.79
3	$10^{4.58}$	1.07	1.16	1.25	5.28	5.42	5.54
4	$10^{5.63}$	7.10	2.38	1.42	20.57	6.72	3.97
5	$10^{5.62}$	18.98	7.29	13.36	81.44	21.42	36.69
6	$10^{2.39}$	5.62	4.75	5.00	270.34	75.98	70.00
7	$10^{2.36}$	2.94	2.55	1.12	65.20	36.97	23.55
8	$10^{6.38}$	5.53	1.19	0.70	16.22	5.35	3.33
9	$10^{5.90}$	4.22	2.73	1.48	11.74	8.54	4.66
10	$10^{3.24}$	1.36	1.32	3.95	19.53	23.22	34.30

Table 3.4: Mean of the respective gains for matrices used in practical applications. The gain values are averaged over 100 trials of random b. The description of the respective matrices can be found in table 3.3

3.4 Inferences

The plots 3.5 to 3.8 showed that the estimated gain was very close to the observed gain, especially for relatively smaller condition numbers. As we increase the condition number, we notice a very prominent trend: the observed gains gradually move below the estimated gain with the increase in condition number. The GMRES gains move further down as compared to CG gains.

3.4.1 Lower observed gain for higher condition number

This can be attributed to the fact that the gain in equation 2.11 assumes the residue of their respective systems as the stopping criteria. For example, the relative residue for a linear system $Ax = b$ and a left preconditioned linear system $MAx = Mb$ are respectively:

$$r = \frac{\|b - Ax\|}{\|b\|}, r_{pre} = \frac{\|M(b - Ax)\|}{\|Mb\|}$$

The upper bounds in equation 2.11 hold when r_{pre} is used for stopping the preconditioned system, whereas we use r everywhere as the stopping criteria. M matrix is supposed to be similar to A^{-1}, hence naturally it'll have a condition number similar to A^{-1} which is equal to the condition number of A. As the condition number increases, so does the difference between r and r_{pre} and hence observed gain deviates from the estimated gain.

3.4.2 Lower gains for GMRES as compared to CG

The reason behind this is also the stopping criteria. We know that GMRES minimizes the 2-norm of the residue. GMRES does so for the system $Ax = b$ but for the preconditioned system, it minimizes r_{pre} instead. Meaning at iteration k, r_{pre} might have gone below the given tolerance, but not r because GMRES was minimizing r_{pre} in this case. The implication is that it takes more number of iterations than expected to converge in the preconditioned case. CG which minimizes the A-norm isn't as affected by the change in parameter for stopping.

3.4.3 Effect of tolerance on Gain

Our estimate does not account for the change in tolerance. Meaning the gain should be the same irrespective of the tolerance. We know that for really high tolerances we will get unexpected results because of limitations in finite arithmetic. Our plots show that for conservative tolerances, the observed gain does not change too much, if at all. Figure 3.9 shows how observed gain varies with tolerance. Even on log scale the gain drops sub-linearly.

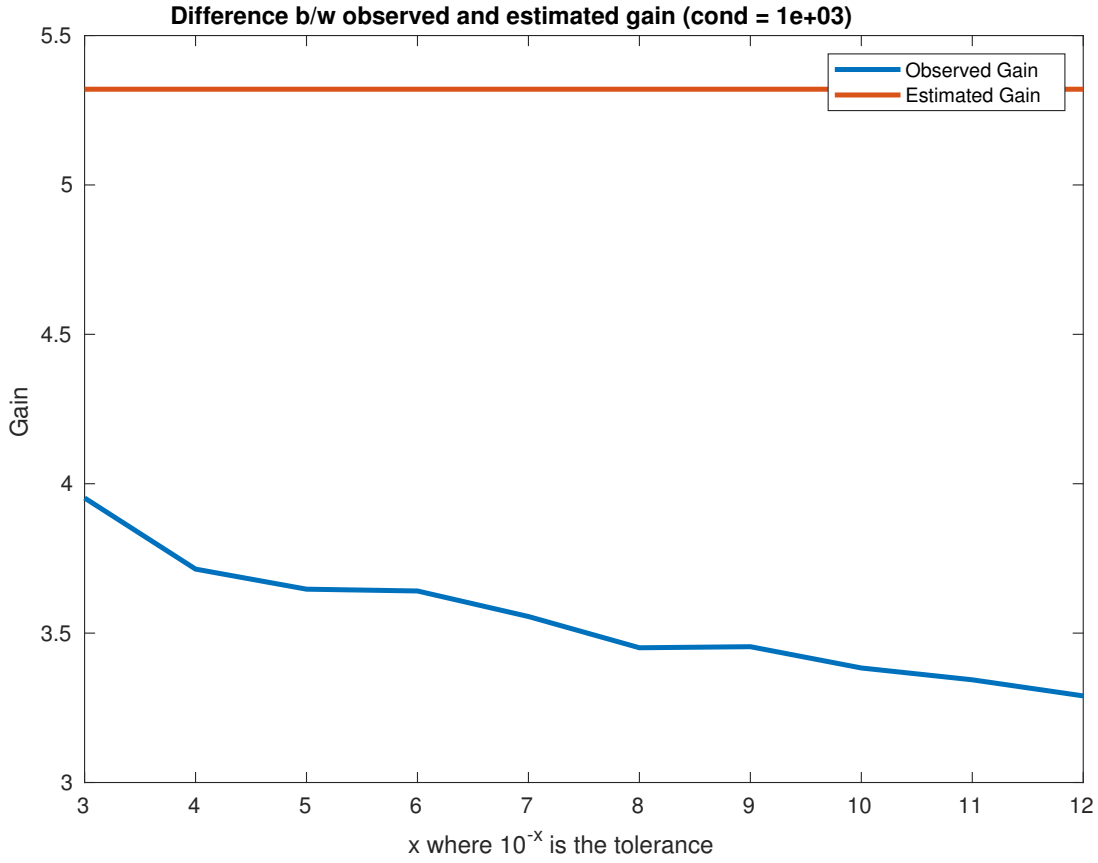

Figure 3.9: The figure shows how the gain behave as tolerance decreases

3.4.4 Effect of N on Gain

We see that in figures 2.3 to 2.6, as we increase N (dimension of the matrix), the iterations taken to converge to a specific tolerance tends to grow with N and eventually becomes asymptotic. Figure 3.10 shows the change in gain with the dimension. We see that for a fixed condition number, we get better gains as N increases. Our estimated gain also shows a similar trend.

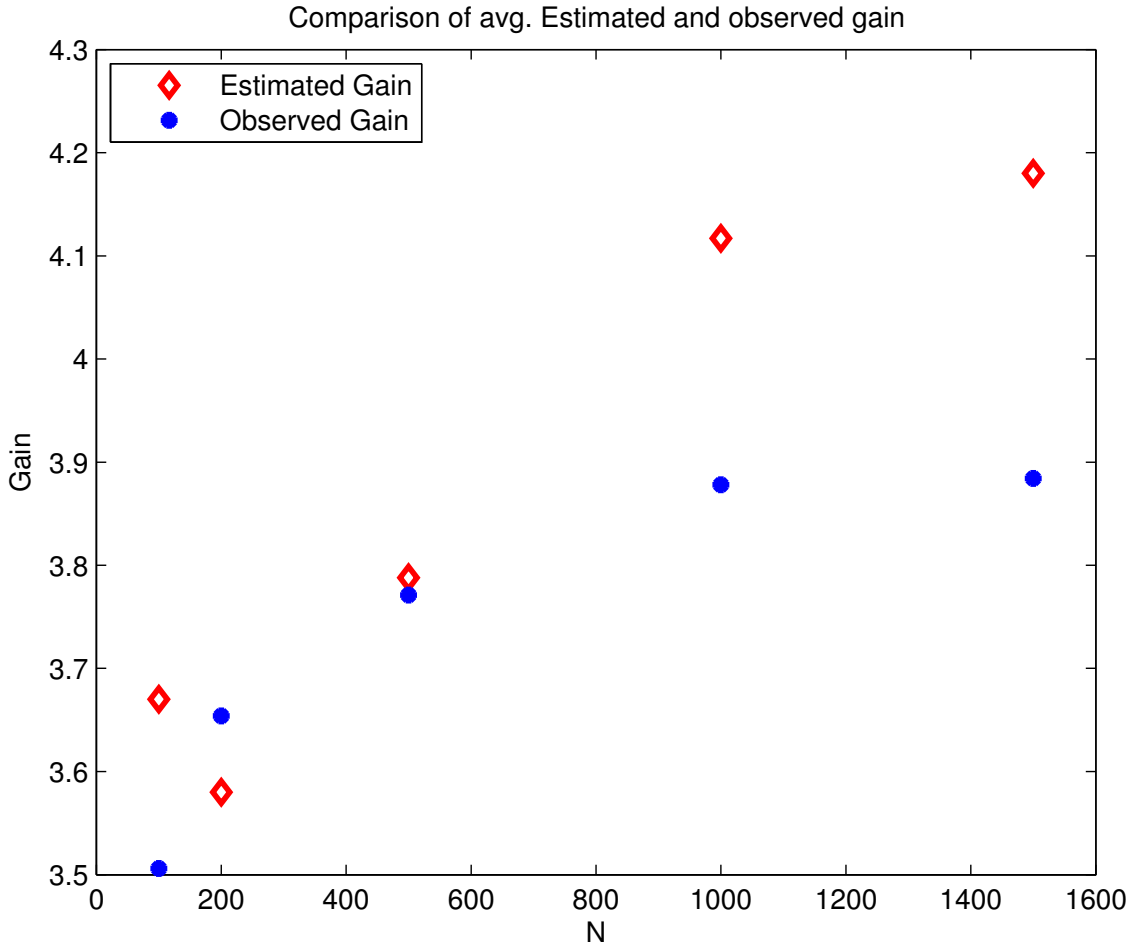

Figure 3.10: The avg estimated and observed gain were averaged (averaged over 1000 matrices) for N=100, 200, 500, 1000, 1500 and plotted for a fixed condition number

3.4.5 Results obtained from real world matrices

In most of the cases in table 3.4, we see trends similar to what we saw for randomly generated matrices. The theoretical gain is close to the estimated gain for matrices with smaller condition number. But as we increase the condition number, the distance between theoretical and actual gains widens as seen for Matrix 2, which has a condition number higher than 10^7. In most cases we do see the CG gain being higher than that of GMRES.

For matrix 6 we see that in the case of Cholesky, we get Theoretical gain of around 270 but the actual gains are much lower. This is because the preconditioned system converges to a

tolerance of $\approx 10^{-15}$ in a single iteration. This has underestimated the actual gain.

Chapter 4

Gains of a preconditioner with and without error estimation

4.1 Problems with the residue as a stopping criteria

For an iterative linear solver, the error at the k_{th} iteration is given by:

$$\epsilon_k = x^* - x_k$$

One would like to stop the iterations when the error ϵ_k reaches a certain tolerance. Unfortunately, x^* is the quantity we are trying to find, meaning we cannot use ϵ_k as our stopping criteria. Instead, residue/relative residue is used to stop more often than not. The residue becoming smaller does imply that we are getting closer to the solution. But the residue being below a certain tolerance does not ensure that the actual error is below that tolerance too [15]. In fact in the worst case, the relative residue and relative error might be off by a factor of κ (Condition number of the matrix), i.e.:

$$\frac{\|r_k\|}{\|b\|\,\kappa} \leq \frac{\|\epsilon_k\|}{\|x^*\|} \leq \frac{\|r_k\|\,\kappa}{\|b\|}$$

Figure 4.1 shows how far the relative residue is from the actual error. They are almost off by a factor of 10^3 for a matrix of condition number 10^4. If we use the residue to stop, then we will certainly stop prematurely and we might not have the accuracy that we actually need. This gave to the rise of error-estimators. These are values that help us approximate ϵ_k at the cost of a fixed number of additional iterations. As seen in the figure, the error estimator does a very good job of estimating the error, in-spite of the high condition number.

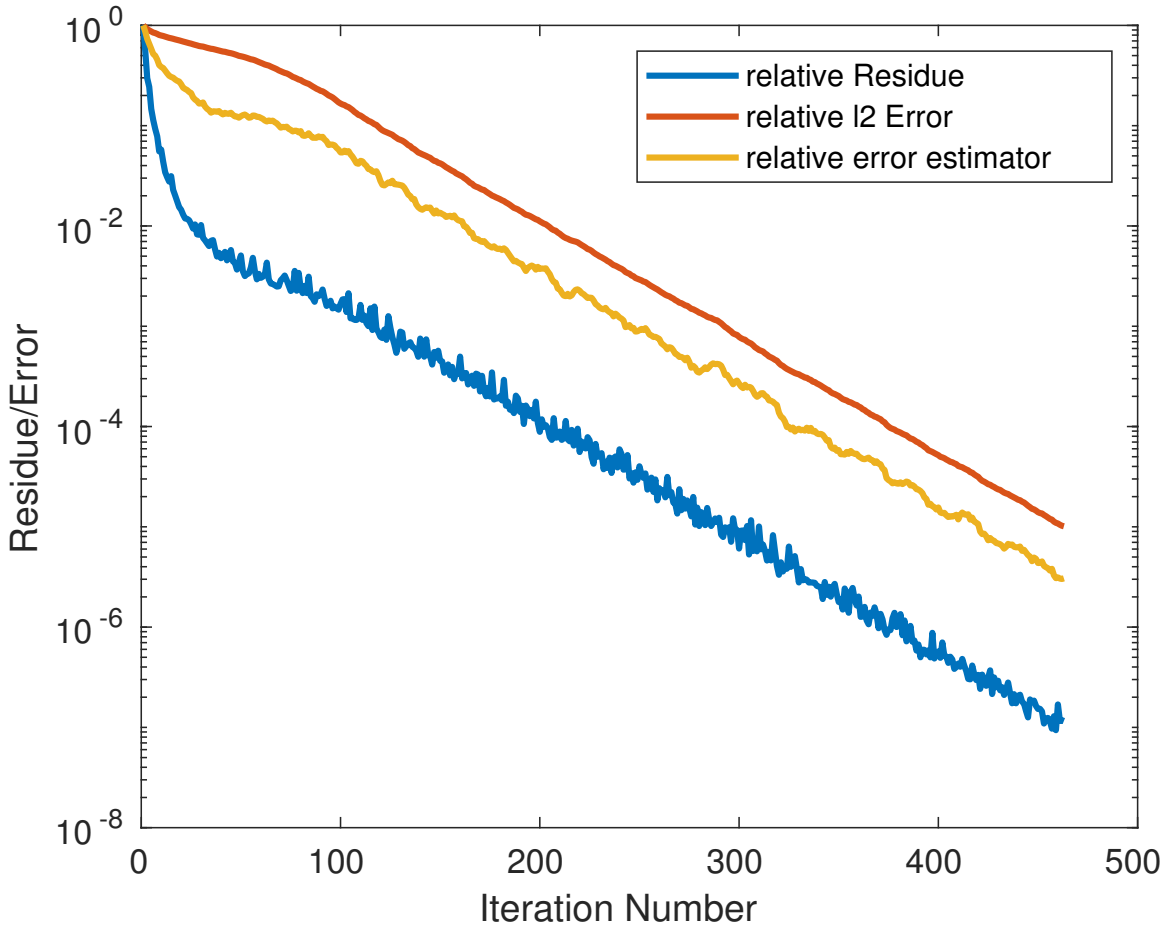

Figure 4.1: Actual and estimated error estimates for a matrix of dimension 1500 and condition number 10^4

For iterative methods, the actual error at k^{th} step is given by $\epsilon_k = (x^* - x_k)$. So we can derive a basic 2-norm error estimator in the following way:

$$\epsilon_k - \epsilon_{k+1} = (x^* - x_k) - (x^* - x_{k+1})$$
$$\implies \epsilon_k - \epsilon_{k+1} = x_{k+1} - x_k$$

Now if we take the telescopic sum of the left hand side over d steps, then:

$$\implies \epsilon_k \approx x_{k+d} - x_k$$

We ignore ϵ_{k+d} from the above expression because for a big enough d, it would be too small as compared to ϵ_k. Another assumption is that the algorithm is converging fast otherwise ϵ_{k+d} and ϵ_k might not be too far apart. d is then called the delay in estimation. Meaning, to get the error estimate at k^{th} step, we would need to run the algorithm for d more steps.

A similar idea is used to derive the error estimators in general. The above is an $\mathcal{O}(N)$ estimator, where N is the dimension of the matrix. Also, within this chapter 'residue' will imply relative residue and 'error' will imply relative error unless specified otherwise.

4.2 Quantifying the uncertainty

As seen in Figure 4.1, the residue introduces some uncertainty with it. The question now arises, if we can enumerate in some way, how well a quantity estimates the error. We use the method used by Puneet [15] to quantify it. We take the difference between the actual error and the estimator divided by the minimum of the two. We call this the Uncertainty Ratio **(U.R)**[15] which is given by:

$$U.R = \frac{1}{m}\sum_{k=1}^{m} \frac{|\text{estimated relative error - relative error}|}{min(\text{estimated relative error, relative error})}$$

The uncertainty is then averaged over the total number of iterations m. It is also worth discussing the minimum in the denominator. The estimator might undercompute or overcompute depending on the error estimator. If the error estimator over-estimates the error, then that leads to greater number of iterations or over-computing and vice versa. The minimum ensures that we take note of the under-computing or over-computing that might happen.

4.3 l2 Error Estimation for Conjugate Gradient

In the method of Conjugate Gradient, the more popularly used measures for stopping are the l2 norm or A-norm of the error. We see in our figure 4.1 that the relative A-norm of the error tends to underestimate the actual error by a huge margin and residue tends to overestimate the same. Thus we look at an error estimator for Conjugate Gradient below:

The error at the k^{th} iteration of the Conjugate Gradient method is given by:

$$\epsilon_k = x^* - x_k$$

Here x^* is the true solution to the equation $Ax = b$ and x_k is the value of x at the k^{th} iteration of the Conjugate Gradient method. Now we look at the difference in errors at successive

iterations:

$$\begin{aligned} \epsilon_k - \epsilon_{k+1} &= (x^* - x_k) - (x * -x_{k+1}) \\ \implies \epsilon_k - \epsilon_{k+1} &= x_{k+1} - x_k \\ \implies \epsilon_k - \epsilon_{k+1} &= \alpha_{k+1} p_{k+1} \end{aligned}$$

p_{k+1} and α_{k+1} are the same as defined in the Conjugate Gradient method. When we take the telescopic sum of these differences over d iterations, then we get:

$$\begin{aligned} \epsilon_k - \epsilon_{k+d} &= \sum_{j=k+1}^{k+d} \alpha_j p_j \\ \implies \epsilon_k &\approx \sum_{j=k+1}^{k+d} \alpha_j p_j \end{aligned} \tag{4.1}$$

We can choose to ignore ϵ_{k+d} since for a large enough d, $\epsilon_k \gg \epsilon_{k+d}$. The term d is referred to as the delay because we need to iterate at least $k + d$ times to get an estimate of error at the k^{th} iteration. We will be using equation (4.1) as our l2 error estimate for all experiments henceforth. The estimator is an $\mathcal{O}(N)$ estimator.

In the preconditioned case, we can use the same estimator because the algorithm is such that it allows us to same p_j that we expected in the non preconditioned case [26].

4.3.1 Results

We took four SPD matrices (1500×1500) with different condition numbers and varied their backward condition number by changing the right hand side b. We then solved them and plotted their uncertainty ratio for the residue as well as the error estimator as seen in Figures 4.2 and 4.4 respectively. We did the same for preconditioned systems as shown in Figures 4.3 and 4.5. The preconditioner used is the Incomplete Cholesky Preconditioner.

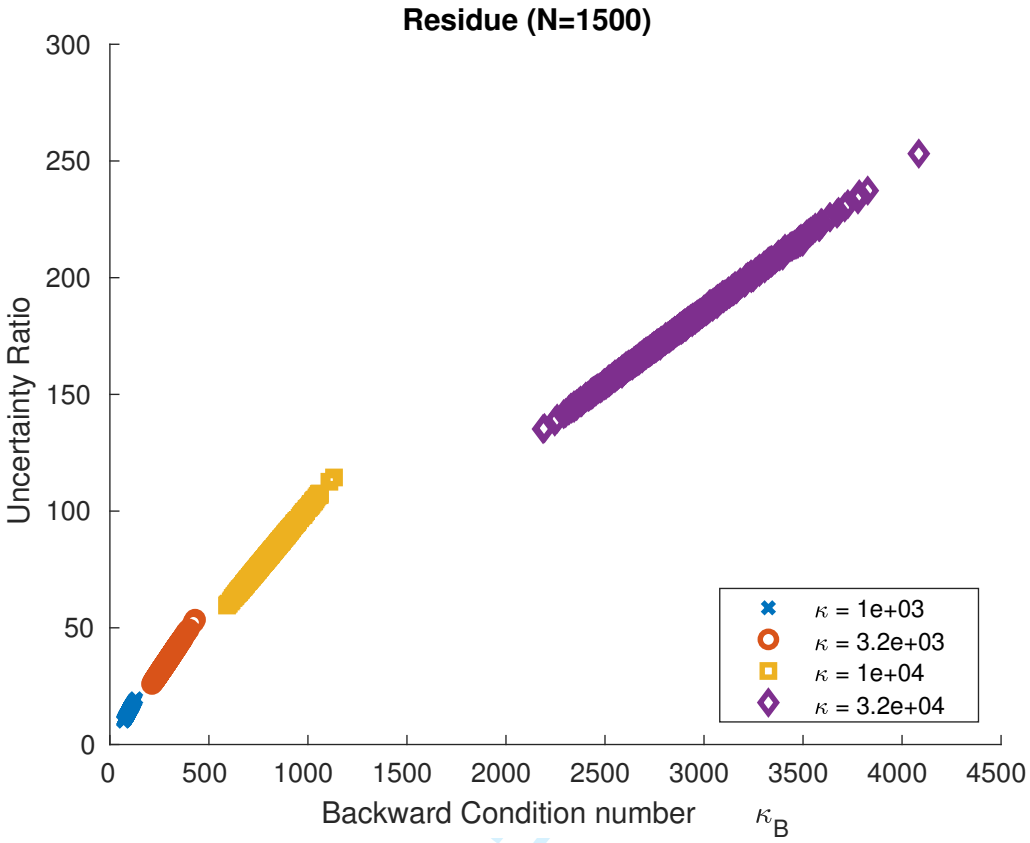

Figure 4.2: Observed uncertainty ratio for the residue of CG

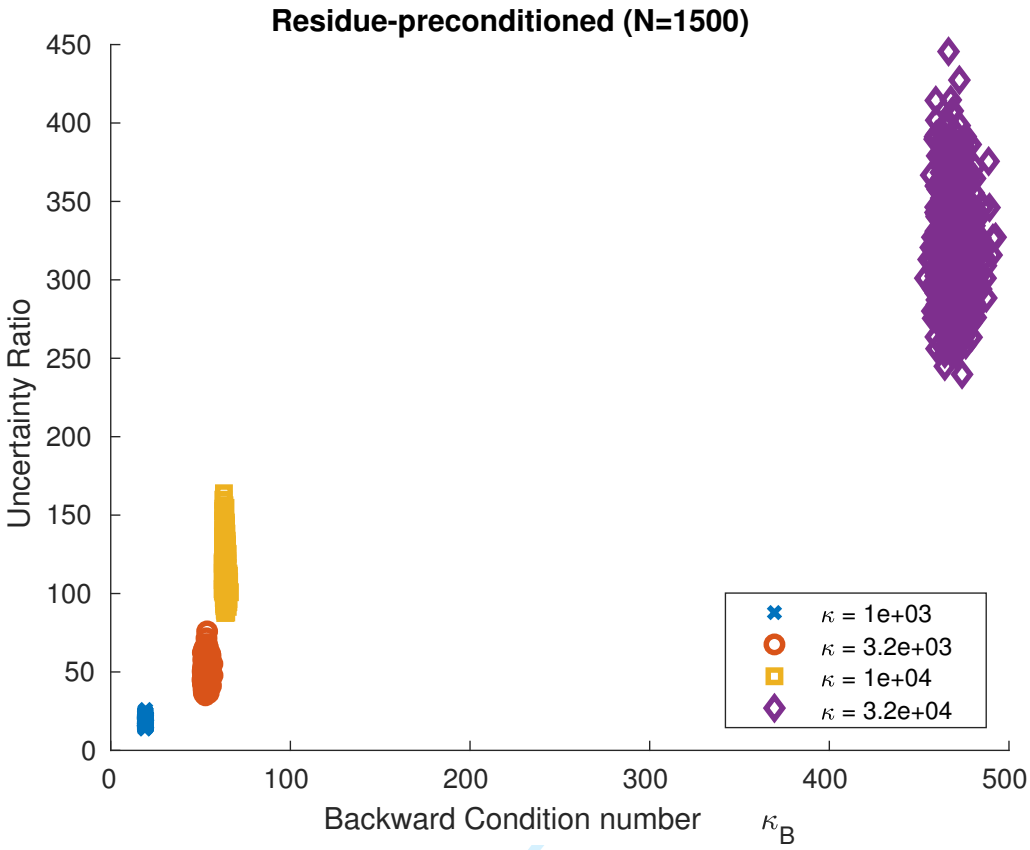

Figure 4.3: Observed uncertainty ratio for the residue of PCG

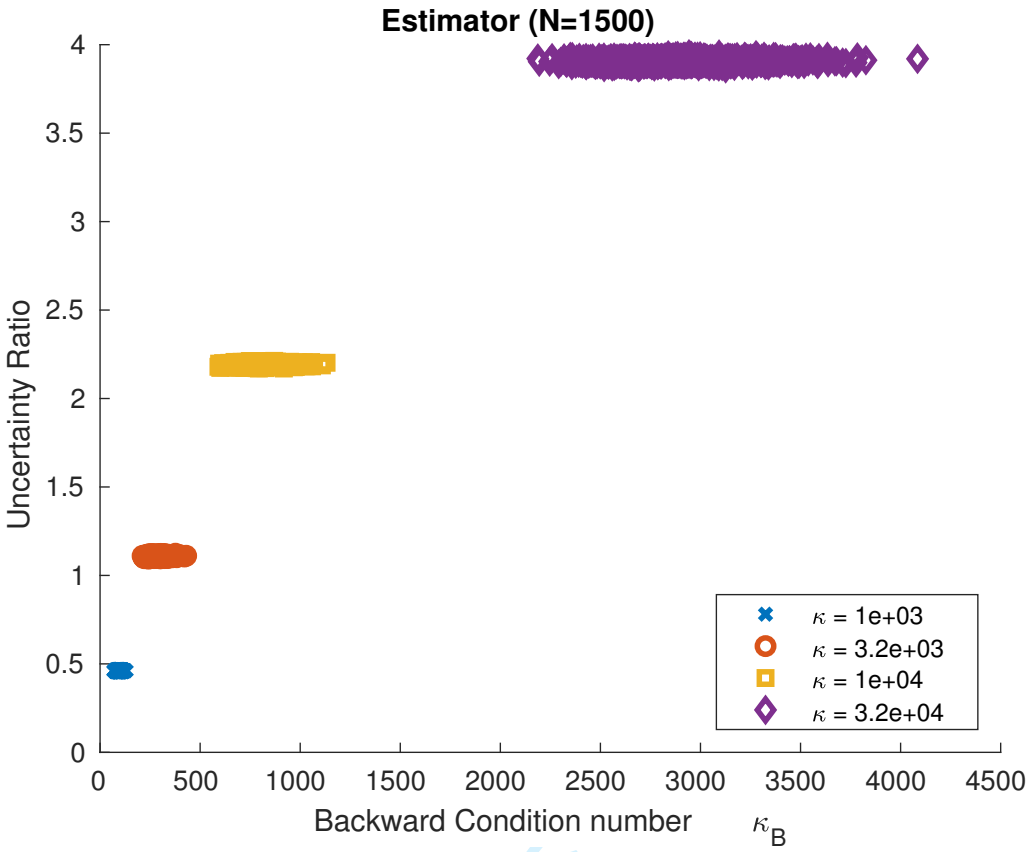

Figure 4.4: Observed uncertainty ratio for the error estimator of CG

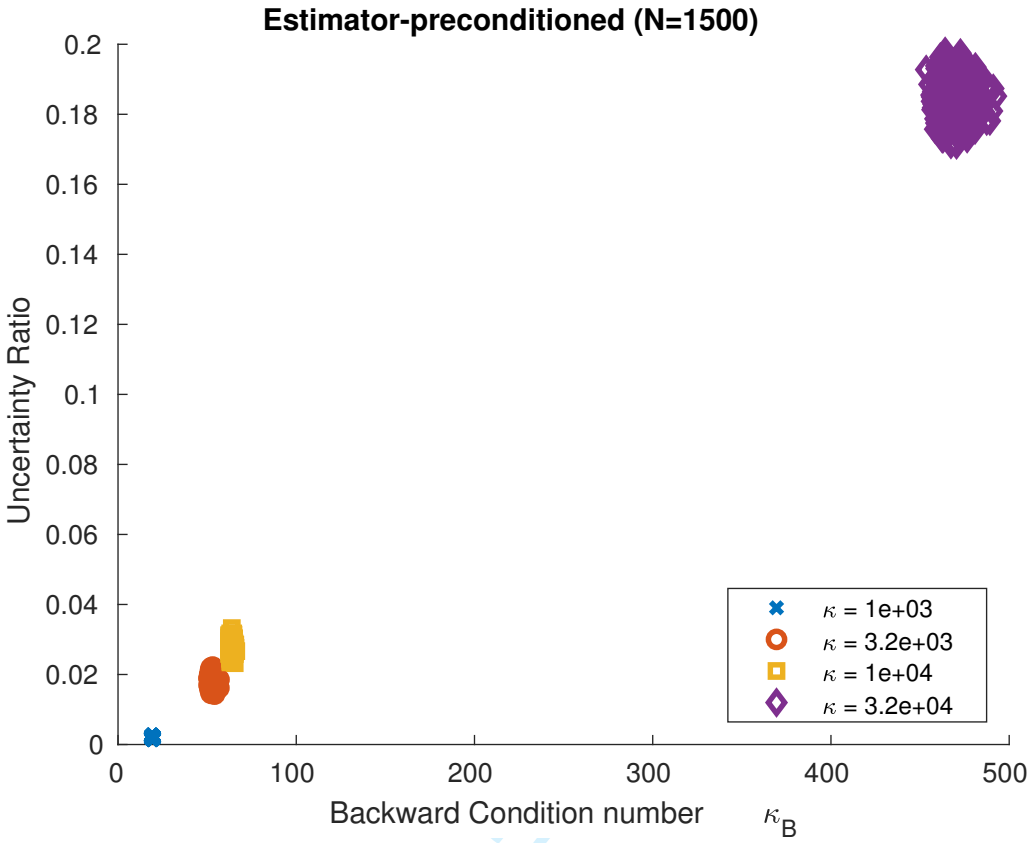

Figure 4.5: Observed uncertainty ratio for the error estimator of PCG

4.3.2 Inference

We notice that the Uncertainty Ratio of residue in both the preconditioned as well as the non-preconditioned case is pretty high. They show significant increase with the increase in condition number. The preconditioned case shows higher uncertainty for residue, mostly due to the fact that preconditioning changes the problem itself. Residue for the new preconditioned system is now $M^{-1}(b-Ax)$ and not $(b-Ax)$ and this produces even more uncertainty. The estimators do well in both the preconditioned and non-preconditioned case, even for high condition numbers. Unlike the residue, uncertainty in estimators does not increase with the change in backward condition number and this further proves its robustness.

The preconditioned error estimator works better than in the general case, very distinctly showing that error estimators with preconditioners is not only a necessity but also yields better results.

4.4 l2 Error Estimation of GMRES

For solving GMRES, we use Puneet's estimator [15] which is a small modification of the Meurant's estimator [20].

GMRES produces an orthonormal set of basis vectors V_k for the k dimensional Krylov subspace at the k^{th} iteration. Let d be the delay. Then the Hessenberg matrix can be split into a block of 4:

$$H_k = \begin{pmatrix} H_{k-d} & W_{k-d} \\ Y_{k-d}^T & \tilde{H}_{k-d} \end{pmatrix}$$

and let

$$\gamma_{k-d} = \frac{h_{k-d+1,k-d}(e_{k-d}, H_{k-d}^{-1} e_1)}{1 - h_{k-d+1,k-d}(e_{k-d}, H_{k-d}^{-1} w_{k-d})}$$

where $w_{k-d} = W_{k-d}\tilde{H}_{k-d}^{-1} e_1$ and

$$\delta_{k+1} = \frac{h_{k+1,k}^2}{1 + h_{k+1,k}^2 t_{kk}}$$

and

$$u_k = \delta_{k+1} t_k$$

where t_k is the final column of $(H_k^T H_k)^{-1}$ and t_{kk} its final element. The resulting error estimate at the $(k-d)^{th}$ iteration proposed by Meurant is:

$$\frac{\chi_{k-d}^2}{\|r_0\|^2} = \gamma_{k-d}^2 \left\| \tilde{H}_{k-d}^{-1} e_1 \right\|^2 + \left\| \gamma_{k-d} H_{k-d}^{-1} w_{k-d} + (e_{k-d}, H_{k-d}^{-1} e_1) u_{k-d} \right\|^2$$

In their paper, they add an extra term to the expression that prevents unstable overshoots in estimation. This estimator is given by:

$$\frac{\chi_{k-d}^2}{\|r_0\|^2} = \left| \gamma_{k-d}^2 \left\| \tilde{H}_{k-d}^{-1} e_1 \right\|^2 + \left\| \gamma_{k-d} H_{k-d}^{-1} w_{k-d} + (e_{k-d}, H_{k-d}^{-1} e_1) u_{k-d} \right\|^2 - \left\| (e_k, H_k^{-1} e_1) u_k \right\|^2 \right|$$

The 'absolute' stops the estimator from becoming negative in non converging situations. It is an $\mathcal{O}(k^2)$ estimator, where k is the current iteration number. Because GMRES is generally used with restarts, this turns out to be a small quantity.

For the preconditioned case, we use $\|L(error\_estimate)\|$ (where L is lower triangular matrix produced in incomplete Cholesky)to calculate the actual error. However, this takes $\mathcal{O}(N^2)$

operations. In order to save operations, one can also find the infinity norm of L and multiply it with the error estimate. This will be less accurate than our plotted estimate but more accurate than the residue nonetheless.

4.4.1 Results

We choose a similar setup as for CG in GMRES too. Figures 4.6 and 4.8 show the residue and error estimator respectively. We did the same for preconditioned GMRES as shown in Figures 4.7 and 4.9.

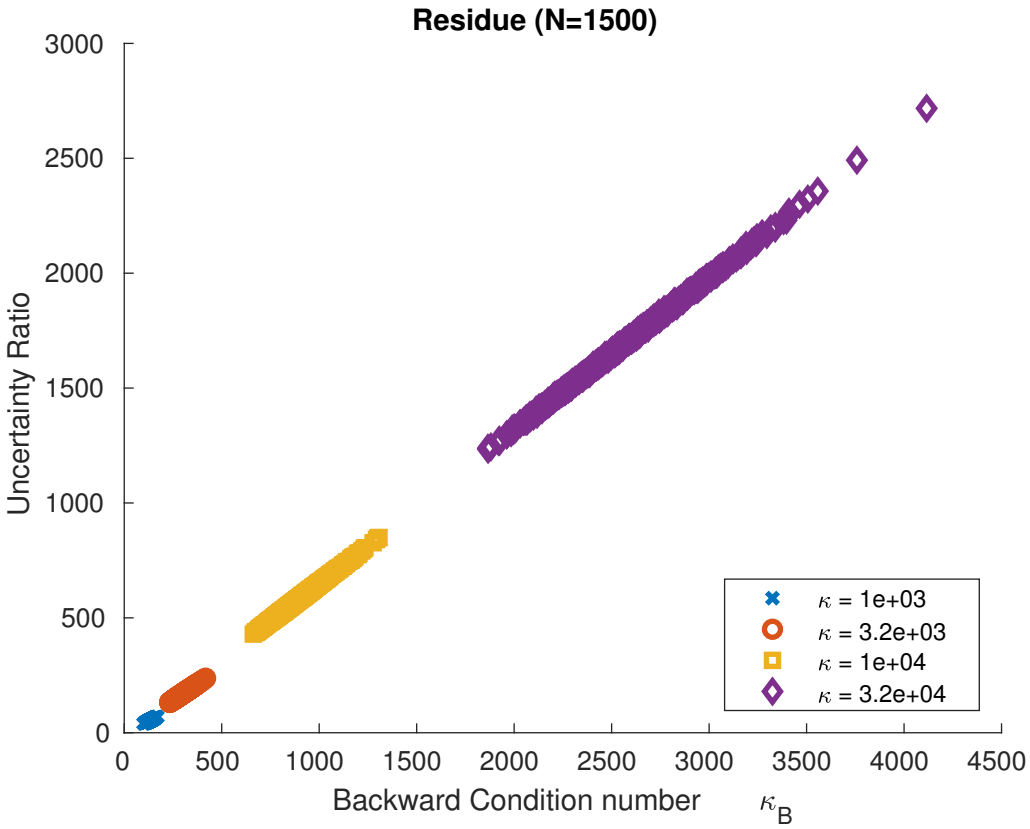

Figure 4.6: Observed uncertainty ratio for the residue of GMRES

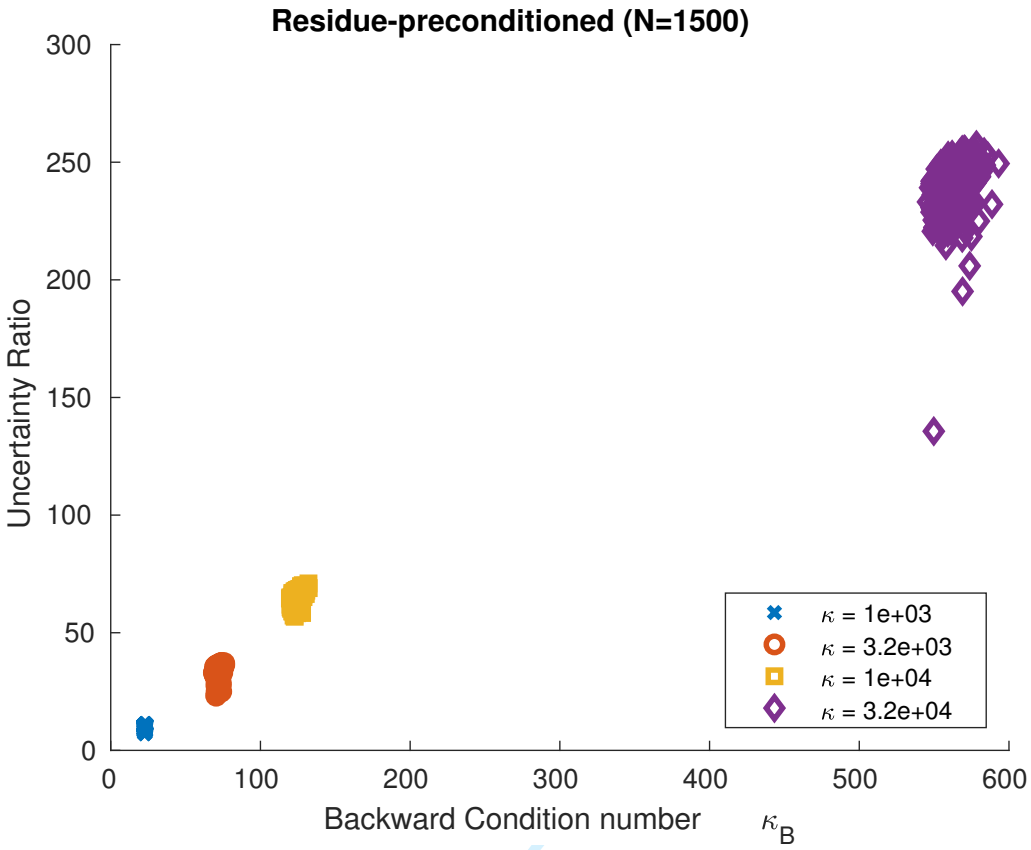

Figure 4.7: Observed uncertainty ratio for the residue of preconditioned GMRES

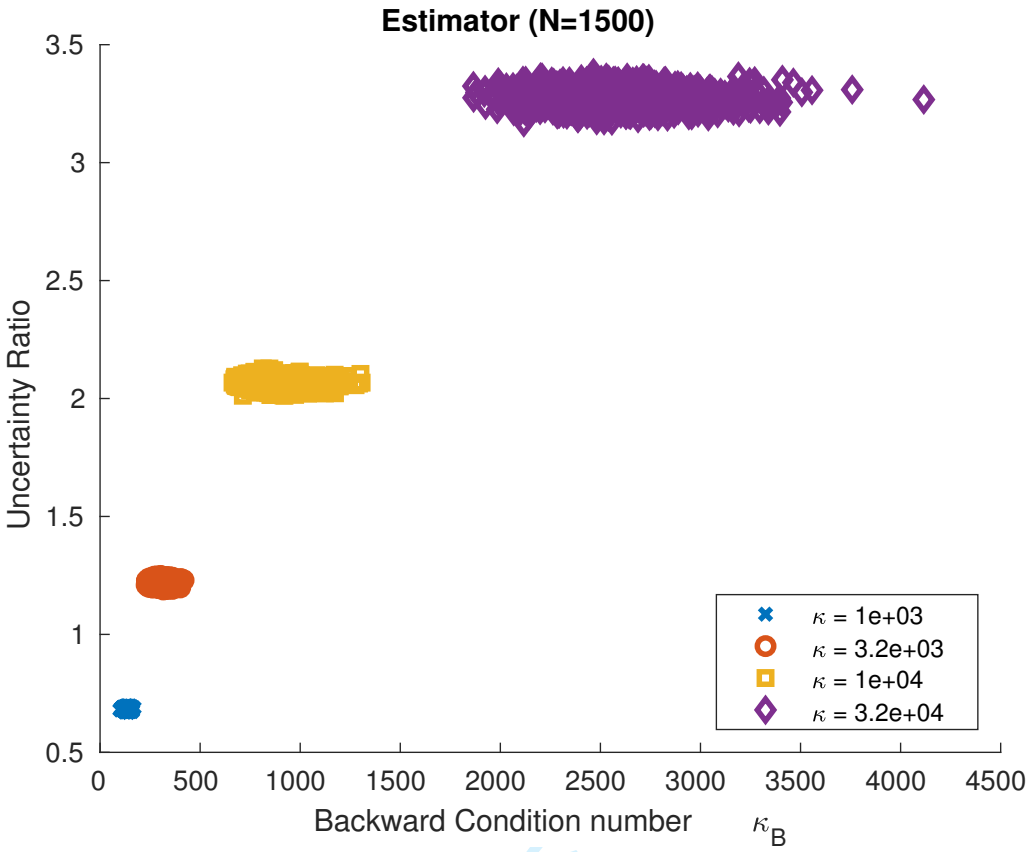

Figure 4.8: Observed uncertainty ratio for the error estimator of GMRES

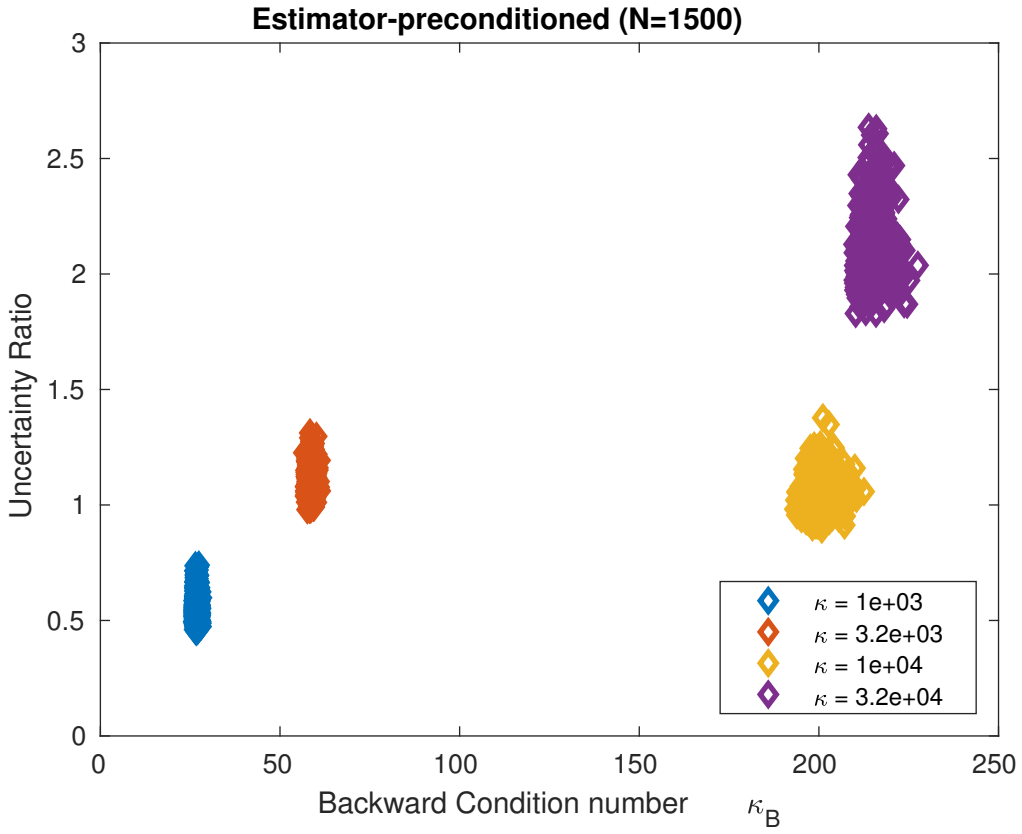

Figure 4.9: Observed uncertainty ratio for the error estimator of preconditioned GMRES

4.4.2 Inference

In the general case, the residue turns out to be a very bad estimator of the error. This is because GMRES minimizes residue causing it to be much lower than the error. Unlike CG, the preconditioned case brings down the uncertainty in residue but dip is not very noteworthy. This means the uncertainty in residue is still significantly high.

The correlation between uncertainty in residue and condition number remains the same as in CG. Higher condition number leads to a higher uncertainty in the residue.

The error estimators work better in both the cases and do not change with the change in backward condition number. This shows that yet again, there is a strong need for error estimators even in preconditioned systems using iterative methods.

Bibliography

[1] William K. Anderson, Stephen Wood, and Kevin E. Jacobson. Node Numbering for Stabilizing Preconditioners Based on Incomplete LU Decomposition. In *AIAA AVIATION 2020 FORUM*, VIRTUAL EVENT, June 2020. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-598-2. doi: 10.2514/6.2020-3022. URL https://arc.aiaa.org/doi/10.2514/6.2020-3022. 10

[2] Hartwig Anzt, Mark Gates, Jack Dongarra, Moritz Kreutzer, Gerhard Wellein, and Martin Khler. Preconditioned Krylov solvers on GPUs. *Parallel Computing*, 68(Supplement C): 32–44, October 2017. ISSN 0167-8191. doi: 10.1016/j.parco.2017.05.006. URL http://www.sciencedirect.com/science/article/pii/S0167819117300777. 9, 11

[3] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel. Adaptively Preconditioned GMRES Algorithms. *SIAM Journal on Scientific Computing*, 20(1):243–269, January 1998. ISSN 1064-8275. doi: 10.1137/S1064827596305258. URL https://epubs.siam.org/doi/abs/10.1137/S1064827596305258. Publisher: Society for Industrial and Applied Mathematics. 7

[4] A. H. Baker, E. R. Jessup, and Tz. V. Kolev. A simple strategy for varying the restart parameter in GMRES(m). *Journal of Computational and Applied Mathematics*, 230(2): 751–761, August 2009. ISSN 0377-0427. doi: 10.1016/j.cam.2009.01.009. URL http://www.sciencedirect.com/science/article/pii/S0377042709000132. 5

[5] Michele Benzi. Preconditioning Techniques for Large Linear Systems: A Survey. *Journal of Computational Physics*, 182(2):418–477, November 2002. ISSN 0021-9991. doi: 10.1006/jcph.2002.7176. URL http://www.sciencedirect.com/science/article/pii/S0021999102971767. 3

[6] Michele Benzi and Miroslav Tma. A robust incomplete factorization preconditioner for positive definite matrices. *Numerical Linear Algebra with Applications*, 10(5-6):385–400,

BIBLIOGRAPHY

2003. ISSN 1099-1506. doi: 10.1002/nla.320. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nla.320. 8, 10, 11

[7] Edmond Chow and Yousef Saad. Experimental study of ILU preconditioners for indefinite matrices. *Journal of Computational and Applied Mathematics*, 86(2):387–414, December 1997. ISSN 03770427. doi: 10.1016/S0377-0427(97)00171-4. URL http://linkinghub.elsevier.com/retrieve/pii/S0377042797001714. 10

[8] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson. An Estimate for the Condition Number of a Matrix. *SIAM Journal on Numerical Analysis*, 16(2):368–375, April 1979. ISSN 0036-1429. doi: 10.1137/0716029. URL https://epubs.siam.org/doi/abs/10.1137/0716029. Publisher: Society for Industrial and Applied Mathematics. 6

[9] Timothy A. Davis and Yifan Hu. The university of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1):1:1–1:25, December 2011. ISSN 0098-3500. doi: 10.1145/2049662.2049663. URL https://doi.org/10.1145/2049662.2049663. 22

[10] E. F. D'Azevedo, P. A. Forsyth, and Wei-Pai Tang. Towards a cost-effective ILU preconditioner with high level fill. *BIT Numerical Mathematics*, 32(3):442–463, September 1992. ISSN 1572-9125. doi: 10.1007/BF02074880. URL https://doi.org/10.1007/BF02074880. 10

[11] Iain S. Duff and Grard A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT Numerical Mathematics*, 29(4):635–657, December 1989. ISSN 1572-9125. doi: 10.1007/BF01932738. URL https://doi.org/10.1007/BF01932738. 8

[12] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. JHU Press, December 2012. ISBN 978-1-4214-0859-0. 5

[13] Magnus R Hestenes and Eduard Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. page 28. 4

[14] Ilse C. F. Ipsen and Carl D. Meyer. The Idea Behind Krylov Methods. *The American Mathematical Monthly*, 105(10):889–899, December 1998. ISSN 0002-9890. doi: 10.1080/00029890.1998.12004985. URL https://doi.org/10.1080/00029890.1998.12004985. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00029890.1998.12004985. 3, 5

BIBLIOGRAPHY

[15] Puneet Jain, Krishna Manglani, and Murugesan Venkatapathi. Significance of error estimation in iterative solution of linear systems : estimation algorithms and analysis for CG, Bi-CG and GMRES. *arXiv:1705.08806 [math]*, May 2017. URL http://arxiv.org/abs/1705.08806. arXiv: 1705.08806. 10, 11, 37, 39, 45

[16] Mark T. Jones and Paul E. Plassmann. An improved incomplete Cholesky factorization. *ACM Transactions on Mathematical Software*, 21(1):5–17, March 1995. ISSN 0098-3500. doi: 10.1145/200979.200981. URL https://doi.org/10.1145/200979.200981. 8

[17] David S Kershaw. The incomplete Choleskyconjugate gradient method for the iterative solution of systems of linear equations. *Journal of Computational Physics*, 26(1):43–65, January 1978. ISSN 0021-9991. doi: 10.1016/0021-9991(78)90098-0. URL http://www.sciencedirect.com/science/article/pii/0021999178900980. 8

[18] Andrew V. Knyazev and Ilya Lashuk. Steepest Descent and Conjugate Gradient Methods with Variable Preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 29 (4):1267–1280, December 2007. ISSN 0895-4798. doi: 10.1137/060675290. URL https://epubs.siam.org/doi/10.1137/060675290. Publisher: Society for Industrial and Applied Mathematics. 7

[19] F.-H. Lee, K. K. Phoon, K. C. Lim, and S. H. Chan. Performance of Jacobi preconditioning in Krylov subspace solution of finite element equations. *International Journal for Numerical and Analytical Methods in Geomechanics*, 26(4):341–372, 2002. ISSN 1096-9853. doi: 10.1002/nag.204. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nag.204. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nag.204. 8

[20] Grard Meurant. Estimates of the Norm of the Error in Solving Linear Systems with FOM and GMRES. *SIAM J. Scientific Computing*, 33:2686–2705, January 2011. doi: 10.1137/100795565. 45

[21] C. C. Paige and M. A. Saunders. Solution of Sparse Indefinite Systems of Linear Equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, September 1975. ISSN 0036-1429, 1095-7170. doi: 10.1137/0712047. URL http://epubs.siam.org/doi/10.1137/0712047. 5

[22] Anna Pyzara, Beata Bylina, and Jarosaw Bylina. The influence of a matrix condition number on iterative methods' convergence. In *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 459–464, September 2011. ISSN: null. 6

BIBLIOGRAPHY

[23] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003. ISBN 978-0-89871-534-7. 13, 15

[24] Youcef Saad. Preconditioning techniques for nonsymmetric and indefinite linear systems. *Journal of Computational and Applied Mathematics*, 24(1):89–105, November 1988. ISSN 0377-0427. doi: 10.1016/0377-0427(88)90345-7. URL http://www.sciencedirect.com/science/article/pii/0377042788903457. 10

[25] Youcef Saad and Martin H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, July 1986. ISSN 0196-5204, 2168-3417. doi: 10.1137/0907058. URL http://epubs.siam.org/doi/10.1137/0907058. 5

[26] Zdenk Strako and Petr Tich. Error Estimation in Preconditioned Conjugate Gradients. *BIT Numerical Mathematics*, 45(4):789–817, December 2005. ISSN 1572-9125. doi: 10.1007/s10543-005-0032-1. URL https://doi.org/10.1007/s10543-005-0032-1. 40

[27] Lloyd N. Trefethen and David Bau III. *Numerical Linear Algebra*. SIAM, June 1997. ISBN 978-0-89871-361-9. Google-Books-ID: 4Mou5YpRD_kC. 1, 2

[28] Qingqing Zheng, Yuanzhe Xi, and Yousef Saad. Multicolor lowrank preconditioner for general sparse linear systems. *Numerical Linear Algebra with Applications*, 27(4), August 2020. ISSN 1070-5325, 1099-1506. doi: 10.1002/nla.2316. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nla.2316. 10