GMRES-based Iterative Refinement for Solving Linear Systems

Shutong Wu[*], Valerie Vreugdenhil[†], Gaofeng Zhou[‡]

April 20, 2024

[*]wu867@mcmaster.ca
[†]vreugdev@mcmaster.ca
[‡]zhoug28@mcmaster.ca

Contents

1 Introduction

In the realm of scientific computing, solving linear systems $Ax = b$ with high efficiency and performance is of paramount importance, particularly in the context of large-scale computing applications. Iterative Refinement (IR), as a long-studied solver to linear equations, enhances the precision of solutions obtained by initial approximations in an iterative manner, and has proven essential in a mixed precision setting. Generalized Minimal Residual Method (GMRES) and LU Decomposition (LU) represent two foundational approaches adapted with IR to form GMRES-IR and LU-IR, respectively. LU-IR applies IR to the results of LU factorizations, offering a direct solution pathway that is often faster but may struggle with error propagation in lower precision environments. GMRES-IR combines the robustness of GMRES with the flexible scheme of mixed-precision IR, making it suitable for solving ill-conditioned systems with stability and efficiency. This report delves into these methods, particularly comparing GMRES-IR and LU-IR, to discern their effectiveness in terms of accuracy, convergence, and computational performance within mixed precision frameworks.

1.1 Iterative Refinement

Iterative Refinement (IR) is a technique used to enhance the accuracy of a solution to a system of linear equations in an iterative manner. The core idea behind IR is to iteratively correct residual errors r in a preliminary solution through less computationally intensive methods, thereby improving both accuracy and performance. IR has proven itself particularly useful in mixed precision contexts.

Consider a linear system represented by:

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

where A is a non-singular matrix, x is the vector of unknowns, and b is the known right-hand side vector. The iterative refinement process begins with an initial approximate solution x_0, which may be obtained by any direct or indirect solver (e.g. with LU factorization). The refinement process then proceeds by identifying and correcting the residual error in the solution repeatedly until certain tolerance condition is met. The residual error r_i in round i is computed by:

$$r_i = b - Ax_{i-1} \tag{2}$$

To improve the solution, a correction d_i is obtained by solving:

$$Ad_i = r_i \tag{3}$$

Adding the correction term d_i to the accumulative solution x_{i-1}, we refine the solution:

$$x_i = x_{i-1} + d_i \tag{4}$$

The whole process can be summarized as the below algorithm.

Algorithm 1 IR. $A \in \mathbb{R}^{n\times n}$ is nonsingular, $b \in \mathbb{R}^n$

```
1: Compute initial approximation $x_0$
2: for $i = 1, 2, \ldots, i_{\max}$ or until convergence do
3:     Compute the residual $r_i = b - Ax_{i-1}$
4:     Solve the correction equation $Ad_i = r_i$ using a suitable solver
5:     Update the solution $x_i = x_{i-1} + d_i$
6: end for
7: return $x_i$ as the refined solution
```

Traditionally, LU factorization is adopted as the solver for x_0 (line 1) and corrections d_i (line 4). This gives LU-IR. Detailed analysis of LU-IR is given in 1.4.

1.2 GMRES

Generalized Minimal Residual Method (GMRES) and its variants are advanced numerical techniques for solving non-symmetric linear systems based on iterative refinement (IR). This method is particularly useful for large or ill-conditioned matrices where direct methods are inefficient or impractical.

As an iterative method for solving non-symmetric linear systems, GMRES seeks to minimize the residual over a Krylov subspace generated by the matrix and the residual vector. The main steps in a GMRES iteration include[4]:

- Using the Arnoldi process to construct an orthogonal basis for the Krylov subspace
- Updating the QR factorization of the resulting Hessenberg matrix
- Solving the resulting least squares problem to update the solution

Consider a linear system represented by:

$$Ax = b \tag{5}$$

Assume that $\mathbf{A}$ is a real nonsingular $\mathbf{N}$ by $\mathbf{N}$ matrix and b is a real vector. Given an initial guess x_0 for the solution, the algorithm generates approximate solutions $x_n, n = 1, 2, ..., \mathbf{N}$ from the linear variety

$$x_0 + \mathbf{K}_n(\mathbf{A}, r_0) \tag{6}$$

minimizing the Euclidean norm of the residual

$$\|\mathbf{b} - A\mathbf{x}_n\| = \min_{\mathbf{u}\in\mathbf{x}_0+\mathcal{K}_n(A,r_0)} \|\mathbf{b} - A\mathbf{u}\|, \tag{7}$$

where $r_0 = b - \mathbf{A}x_0$ is the initial residual and $\mathrm{K}_n(\mathbf{A}, r_0)$ is the $n-th$ Krylov subspace generated by $\mathbf{A}$, r_0

$$K_n(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{n-1}r_0\}. \tag{8}$$

Clearly, $r_n \in r_0 + AK_n(A, r_0)$. We call $\mathbf{AK}_n(\mathbf{A}, r_0)$ a Krylov residual subspace.

Such an approximation always exist and is unique, though it can be computed in many different ways.

Most methods for computing the approximation x_n satisfying (7) start by constructing an orthonormal basis, called an Arnoldi basis, for the Krylov subspace (8). The recurrence for the basis vectors can be written in matrix for as

$$AV_n = V_{n+1}H_{n+1,n} \tag{9}$$

where V_{n+1} is the N by $(n+1)$ matrix with the orthonormal basis vectors $V_1, V_2, \ldots, V_{n+1}$ as its columns and $H_{n+1,n}$ is the $(n+1)$ by n upper Hesenberg matrix of the orthogonalization (and normalization) coefficients, $n < N$. The approximate solution x_n is then taken to be of the form $x_n = x_0 + V_n y_n$, where y_n is chosen to minimize

$$\|\mathbf{b} - A\mathbf{x}_n\| = \|\mathbf{r}_0 - AV_n\mathbf{y}_n\| = \|V_{n+1}(\mathbf{e}_1 - H_{n+1,n}\mathbf{y}_n)\| = \|(\mathbf{e}_1 - H_{n+1,n}\mathbf{y}_n)\|. \tag{10}$$

The problem of solving approximately the original N-dimensional system $Ax = b$ is thus transformed to the n-dimensional least squares problem

$$\|\mathbf{e}_1 - H_{n+1,n}\mathbf{y}_n\| = \min_{\mathbf{y}} \|\mathbf{e}_1 - H_{n+1,n}\mathbf{y}\|, \quad \mathbf{x}_n = \mathbf{x}_0 + V_n\mathbf{y}_n. \tag{11}$$

We call $b - Ax_n$ a true residual and $\rho e_1 - H_{n+1,n}y_n$ an Arnoldi residual, where $\rho = \|\mathbf{r}_0\|$.

In Algorithm 1, if GMRES is used as solver for x_0 and corrections d_i, the algorithm is called GMRES-based IR (GMRES-IR). Detailed analysis of GMRES-IR is given in 1.4.

1.3 Mixed Precision Computing

Mixed precision computing has emerged as a highly effective strategy in numerical computing, particularly for enhancing the performance and efficiency of iterative refinement (IR) techniques in solving systems of linear equations. This approach leverages computational hardware optimally by using different precision levels for different parts of the computation, thus balancing between performance and overall accuracy. The core idea behind mixed precision computing is to utilize lower precision for less critical computations and higher precision for more sensitive operations, thus 'saving' precision where it is not needed and 'spending' it where it is essential.

With the advent of modern processors capable of efficiently handling varying precisions—from single (FP32) and double (FP64) to even lower precision formats like half-precision (FP16), mixed precision strategies have become increasingly relevant.

In mixed precision IR, the initial solution and correction computation might be performed in lower precision to exploit faster arithmetic operations. While for critical steps, particularly the residual calculation, it may utilize higher precision to ensure that the refinement process robustly improves the solution without being affected by numerical instability.

To denote the precision levels used in mixed precision computing, we use unit round-off u, also called machine epsilon, as a label for the different precisions. For example, u_{16} denote the use of half-precision (FP16)($\approx 2^{-11}$).

1.4 Comparative Analysis of GMRES-IR and LU-IR

This section conducts a comparative analysis of GMRES-based IR(GMRES-IR) and LU-based IR(LU-IR) in terms of accuracy, convergence, and performance in a mixed-precision setting.

1.4.1 LU-IR

Iterative refinement has traditionally used LU factorization as the inner solver. A basic iterative refinement algorithm using LU factorization can be described as follows:

Algorithm 2 LU-IR. $A \in \mathbb{R}^{n\times n}$ is nonsingular and $b \in \mathbb{R}^n$. Three precisions are used, satisfying $u_r \leq u \leq u_l$.

```
1: Compute the factorization $A = LU$ in precision $u_l$.
2: Solve $LUx_1 = b$ by substitution in precision $u_l$.
3: for $i = 1 : i_{\max}$ or until converged do
4:     Compute $r_i = b - Ax_i$ in precision $u_r$.
5:     Solve $LUd_i = r_i$ by substitution in precision $u_l$.
6:     Update $x_{i+1} = x_i + d_i$ in precision $u$.
7: end for
```

Here the most computational heavy step is the LU factorization, which costs $O(n^3)$ flops. The substitutions on line 5 costs in total $O(Kn^2)$ flops, which is negligible compared to LU factorization for large n and reasonable K (K is the total number of iterations). So by adopting a lower precision for u_l, the computational cost can be greatly reduced. With half-precision availability (u_{16}), the potential speedup with $u = u_r = u_{64}$ and $u_l = u_{16}$ on an NVIDIA P100 GPU can reach 2.7[2].

The convergence of LU-IR depends on various factors, including numerical stability of the LU decomposition, precision scheme, and the condition of A. Specifically, with $\kappa(A)u_l \ll 1$ and a numerically stable LU decomposition, convergence is guaranteed. However, when lower precision is applied (especially in half-precision schemes), the requirement for $\kappa(A)$ tightens. For instance, with fp16, the unit round-off is $u_{16} = u_l \approx 2^{-11}$, rendering $\kappa(A) \ll 2000$. With bfloat16, the unit round-off is $u_{16} = u_l \approx 2^{-8}$, giving an even tighter bound of $\kappa(A) \ll 300$[1]. This marks one limitation of LU-based IR, which is the convergence heavily relies on the system condition and unit round-off. Especially in a half-precision computing setting, this condition limitation of LU-IR render itself a strict and picky solver.

The above algorithm achieves a relative error bounded approximately by $3n\||A^{-1}||\hat{L}||\hat{U}|\|u_l$[1].

1.4.2 GMRES-IR

The GMRES-based iterative refinement (GMRES-IR) in a mixed-precision setting can be described as follows:

Algorithm 3 GMRES-IR. $A \in \mathbb{R}^{n\times n}$ is nonsingular and $b \in \mathbb{R}^n$. Five precisions are used: u_r, u_g, u_p, u, u_l.

```
1: Compute the factorization $A = LU$ in precision $u_l$.
2: Solve $LUx_1 = b$ by substitution in precision $u_l$.
3: for $i = 1 : i_{\max}$ or until converged do
4:     Compute $r_i = b - Ax_i$ in precision $u_r$.
5:     Solve $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$ by GMRES in precision $u_g$, performing the products
       with $U^{-1}L^{-1}A$ in precision $u_p$.
6:     Compute $x_{i+1} = x_i + d_i$ in precision $u$.
7: end for
```

An important feature of GMRES-IR is its applicability to extremely ill-condition matrices.

In Algorithm 3 line 4, denote the relative error:

$$\xi_i = \frac{\|d_i - \hat{d}_i\|_\infty}{\|d_i\|_\infty}$$

and let

$$\mu_i = \frac{\|A(\mathbf{x}_i - \hat{\mathbf{x}}_i)\|_\infty}{\|A\|_\infty \|\mathbf{x}_1 - \hat{\mathbf{x}}_i\|_\infty} \leq 1,$$

$$\phi_i = 2\min(\mathrm{cond}(A), \kappa_\infty(A)\mu_i)u_l + \xi_i,$$

where cond(A) $= \||A^{-1}||A||x|\|_\infty$. As long as ϕ_i is sufficiently less than 1, ξ_i is reduced at each iteration until an iterate $\hat{x}$ is obtained such that:

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \lesssim u + 4p\ cond(A,x)u_r$$

where p is the maximum number of nonzeros in any row of $[A\ b]$, and cond(A,x) is defined as

$$cond(A,x) = \frac{\||A^{-1}||A||x|\|_\infty}{\|x\|_\infty}.$$

This accuracy bound depends on precisions u and u_r, and is independent of precision u_l and x_1. This motivates adopting a higher precision for the residual u_r. It has been experimentally proven that by adopting $u = u_l$ and $u_r = u_g = u_p = u^2$ results in $\phi \approx \xi_i$, and therefore $\xi_i \approx u$ as long as $\kappa(A) \approx u^{-1}$[3]. These results indicate that the error is of the same order as the precision of the computation regardless of the system condition. This is a significant result, as it suggests that systems that are ill-conditioned, which would typically be problematic due to their high sensitivity to errors, can still be solved to a high degree of accuracy using GMRES-IR, with the caveat that a higher precision is used in part of the computation to control the growth of errors. This demonstrates the advantage of GMRES-based IR over LU-based IR, where the requirement for system condition is greatly relaxed with reasonable computational overhead.

In terms of error behavior, one caveat of using GMRES-based method is the possibility of error accumulation inside the subspace iteration. As a Krylov subspace iterative method, round-off errors accumulated along the recurrent chain might result in a considerable deviation from the true value. Here the trade-off is the longer the Krylov subspace that is constructed the more dimensions available and therefore higher accuracy solution and faster convergence, while at the same time it also leads to larger chance of round-off error accumulation as a result of long inner iteration chain. A possible mitigation to such problem is the constant restart, where the Krylov subspace is reset after a preset amount of iterations. In the iterative refinement setting, as GMRES is employeed as a inner solver, returning to the outer loop itself acts as a reset on the subspace. Thus the enforced restart is more likely to be triggered with low or no preconditioning, as in such cases more calculation is carried out inside the Krylov subspace.

In the above algorithm, the preconditioner $U^{-1}L^{-1}$ is used, which costs $O(n^3)$ flops. To avoid such cost, it's also possible to use a less computational expensive preconditioner for GMRES-IR, or even no preconditioner at all. With this tradeoff, larger iteration count is expected, but convergence is still guaranteed with condition $\kappa(A)u_l \ll 1$ met. Examples of less costy preconditioners used in GMRES-IR include incomplete LU factorization, block Jacobi and polynomial preconditioners[5]. With less restrictive preconditioning and system condition requirements, GMRES-IR can choose more flexible precision schemes compared to LU-IR, thus utilizing mixed-precision computing to a higher level.

2 Experiments

2.1 Matrix Generation and Results

Firstly, Functions to generate those matrices to do this experiment were designed as bellow:

Matrix Generation Function:

```
function A = genMatrix(size, cond)
    A = 2*rand(size);
    [u, s, v] = svd(A);
    s = diag(s);
    s = s(1)*( 1-((cond-1)/cond)*(s(1)-s)/(s(1)-s(end))) ;
    s = diag(s);
    A = u*s*v';
end
```

Listing 1: Function to generating matrices

Iteration Refinement Function:

```
    function [x, relres, itcount] = iterref(A,b)
% Solves A*x=b using Gauss elimination and iterative refinement.
% A is a dense, n-by-n double precision matrix and b is an n double
% precision vector.
% The LU factorization of A is computed in single precision and the first
% approximate solution x1 is computed in single precision.
% The residual is evaluated in double precision, and the corrected
   solution
% is in double precision.
%  x contains the computed solution
%  relres(i) contains norm(b-A*x)/norm(b) for the i-th iteration

```

```matlab
[n,~] = size(A);

% to accumulate solution
x = zeros(n,1);

% LU decompoisition
[L, U, P] = lu(single(A),'vector');
opts.LT = true;
opts2.UT = true;

% store b in single precision
r = single(b);

tol = 10*eps;
itcount = 1;
rnorm = norm(r);
bnorm = rnorm;
rnormold = 2*rnorm;
while itcount <= 30 && rnorm>tol*bnorm && rnorm < 0.9*rnormold
  y = linsolve(L, r(P), opts);
  r = linsolve(U, y, opts2);
  x = double(x)+double(r);
  r = b-A*x;
  rnorm = norm(r);
  relres(itcount) = rnorm;
  itcount = itcount +1;
end
end
```

Listing 2: Function to do iteration refinement

And then,Several testing matrices were taken as bellow to do the experiment:

Small size matrices:

```
cond = [1e2, 1e3, 1e4, 1e5, 1e6, 1e7, 1e8, 1e9];
matrices = cell(size(cond,2));
for i=1:size(cond,2)
    matrices{i} = genMatrix(500, cond(i));
end

relresgm = zeros(size(matrices,2),1);
relreslu = zeros(size(matrices,2),1);
itergm = zeros(size(matrices,2),1);
iterlu = zeros(size(matrices,2),1);
timegm = zeros(size(matrices,2),1);
timelu = zeros(size(matrices,2),1);

for i = 1:size(matrices,2)
    b = rand(size(matrices{i},1),1);
    tic
    [x, flag, relres, iter] = gmres(matrices{i}, b, [], 1e-6, 30);
    timegm(i) = toc;
    relresgm(i) = relres;
    flags(i) = flag;
    itergm(i) = iter(2);

    tic
    [x, relres, iter] = iterref(matrices{i}, b);
    timelu(i) = toc;
    relreslu(i) = relres(size(relres,2));
    iterlu(i) = iter;
end

```

```
figure(1)
loglog(cond, relreslu);
title("luir: Relative Residual vs Condition Number (n = 500)")
xlabel("Condition Number")
ylabel("Relative Residual")
saveas(gcf,'luir_acc_smat.png')
figure(2)
loglog(cond, relresgm);
title("gmres: Relative Residual vs Condition Number (n = 500)")
xlabel("Condition Number")
ylabel("Relative Residual")
saveas(gcf,'gmres_acc_smat.png')

figure(3)
semilogx(cond, iterlu, cond, itergm);
title("Iterations vs Condition Number (n = 500)")
xlabel("Condition Number")
ylabel("Number of Iterations")
legend("luir", "gmres")
saveas(gcf,'iter_smat.png')

figure(4)
semilogx(cond, timelu, cond, timegm);
title("Time vs Condition Number (n = 500)")
xlabel("Condition Number")
ylabel("Time (seconds)")
legend("luir", "gmres")
saveas(gcf,'time_smat.png')
```

Listing 3: Small size mtrix

With these small matrices, Results were produced as below:

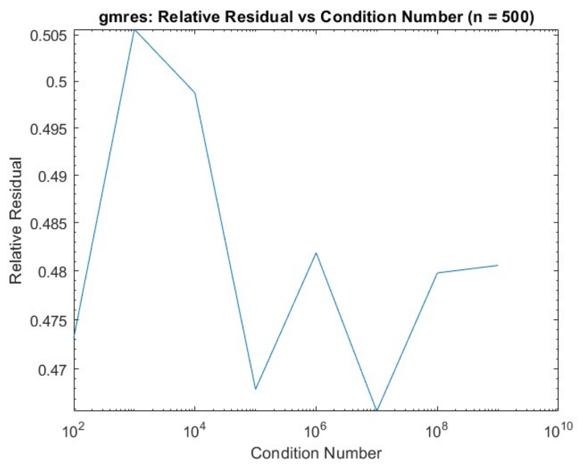

Figure 1: Relative residual with condition number with GMRES

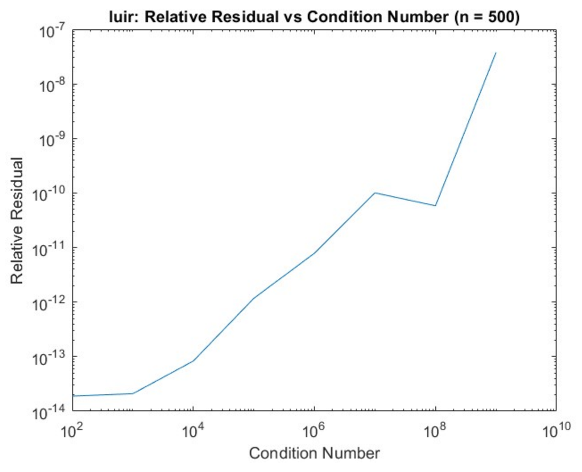

Figure 2: Relative residual with condition number with LU-IR.

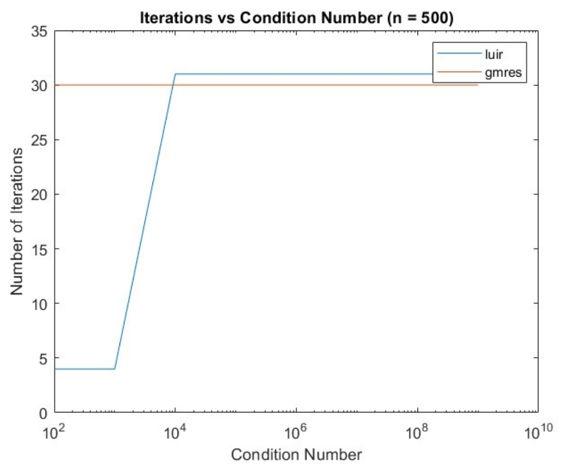

Figure 3: Iterations with condition number

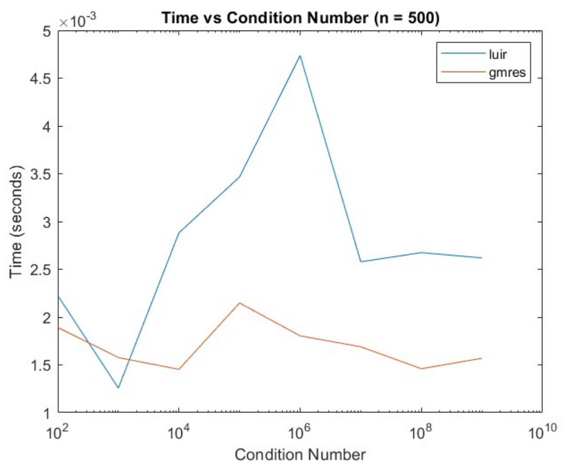

Figure 4: Time with condition number.

Large size matrices:

```
cond = [1e2, 1e3, 1e4, 1e5, 1e6, 1e7, 1e8, 1e9];
matrices = cell(size(cond,2));
for i=1:size(cond,2)
    matrices{i} = genMatrix(5000, cond(i));
end

relresgm = zeros(size(matrices,2),1);
relreslu = zeros(size(matrices,2),1);
itergm = zeros(size(matrices,2),1);
iterlu = zeros(size(matrices,2),1);
timegm = zeros(size(matrices,2),1);
timelu = zeros(size(matrices,2),1);
flags = zeros(size(matrices,2),1);

for i = 1:size(matrices,2)
    b = rand(size(matrices{i},1),1);
    tic
    [x, flag, relres, iter] = gmres(matrices{i}, b, [], 1e-6, 30);
    timegm(i) = toc;
    relresgm(i) = relres;
    flags(i) = flag;
    itergm(i) = iter(2);

    tic
    [x, relres, iter] = iterref(matrices{i}, b);
    timelu(i) = toc;
    relreslu(i) = relres(size(relres,2));
    iterlu(i) = iter;
end

```

```
figure(1)
loglog(cond, relreslu);
title("luir: Relative Residual vs Condition Number (n = 5000)")
xlabel("Condition Number")
ylabel("Relative Residual")
saveas(gcf,'luir_acc_lmat.png')
figure(2)
loglog(cond, relresgm);
title("gmres: Relative Residual vs Condition Number (n = 5000)")
xlabel("Condition Number")
ylabel("Relative Residual")
saveas(gcf,'gmres_acc_lmat.png')

figure(3)
semilogx(cond, iterlu, cond, itergm);
title("Iterations vs Condition Number (n = 5000)")
xlabel("Condition Number")
ylabel("Number of Iterations")
legend("luir", "gmres")
saveas(gcf,'iter_lmat.png')

figure(4)
semilogx(cond, timelu, cond, timegm);
title("Time vs Condition Number (n = 5000)")
xlabel("Condition Number")
ylabel("Time (seconds)")
legend("luir", "gmres")
saveas(gcf,'time_lmat.png')
```

Listing 4: Large size matrix

With large matrices as above, Results were produced as below:

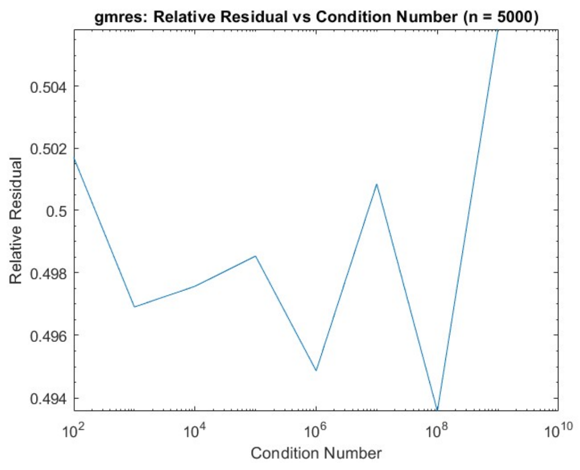

Figure 5: Relative residual with condition number with GMRES

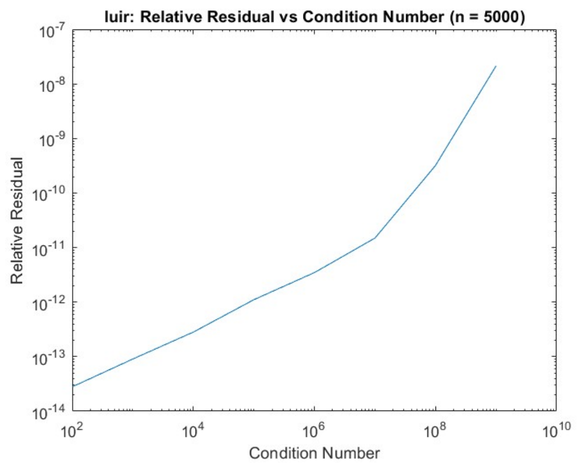

Figure 6: Relative residual with condition number with LU-IR.

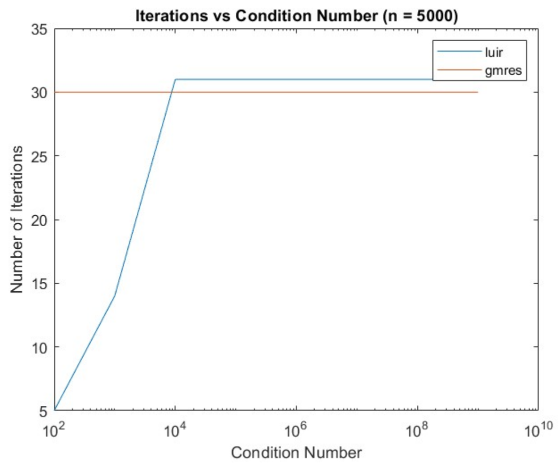

Figure 7: Iterations with condition number.

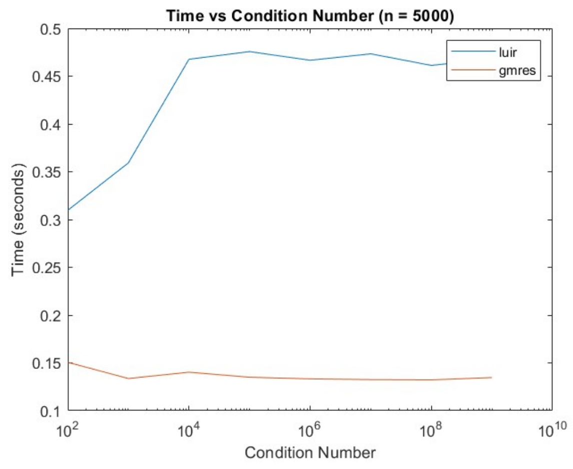

Figure 8: Time with condition number.

Matrices with high condition number:

```matlab
sizes = [10,100,500,1000, 5000];
matrices = cell(size(sizes,2));
for i=1:size(sizes,2)
    matrices{i} = genMatrix(sizes(i), 1e9);
end

relresgm = zeros(size(matrices,2),1);
relreslu = zeros(size(matrices,2),1);
itergm = zeros(size(matrices,2),1);
iterlu = zeros(size(matrices,2),1);
timegm = zeros(size(matrices,2),1);
timelu = zeros(size(matrices,2),1);
flags = zeros(size(matrices,2),1);

for i = 1:size(matrices,2)
    b = rand(size(matrices{i},1),1);
    tic
    [x, flag, relres, iter] = gmres(matrices{i}, b, [], 1e-6, 30);
    timegm(i) = toc;
    relresgm(i) = relres;
    flags(i) = flag;
    itergm(i) = iter(2);

    tic
    [x, relres, iter] = iterref(matrices{i}, b);
    timelu(i) = toc;
    relreslu(i) = relres(size(relres,2));
    iterlu(i) = iter;
end

```

```
figure(1)
semilogy(sizes, relreslu);
title("luir: Relative Residual vs Matrix Size (condition = 1e9)")
xlabel("Matrix Size")
ylabel("Relative Residual")
saveas(gcf,'luir_acc_hcond.png')
figure(2)
semilogy(sizes, relresgm);
title("gmres: Relative Residual vs Matrix Size (condition = 1e9)")
xlabel("Matrix Size")
ylabel("Relative Residual")
saveas(gcf,'gmres_acc_hcond.png')

figure(3)
plot(sizes, iterlu, sizes, itergm);
title("Iterations vs Matrix Size (condition = 1e9)")
xlabel("Matrix Size")
ylabel("Number of Iterations")
legend("luir", "gmres")
saveas(gcf,'iter_hcond.png')

figure(4)
plot(sizes, timelu, sizes, timegm);
title("Time vs Matrix Size (condition = 1e9)")
xlabel("Matrix Size")
ylabel("Time (seconds)")
legend("luir", "gmres")
saveas(gcf,'time_hcond.png')
```

Listing 5: Matrix with high contional number

With matrices with high condition number, Results were produced as below:

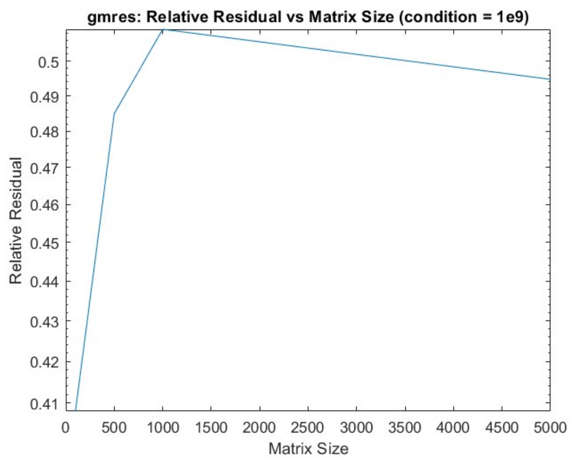

Figure 9: Relative residual with size of matrix with GMRES

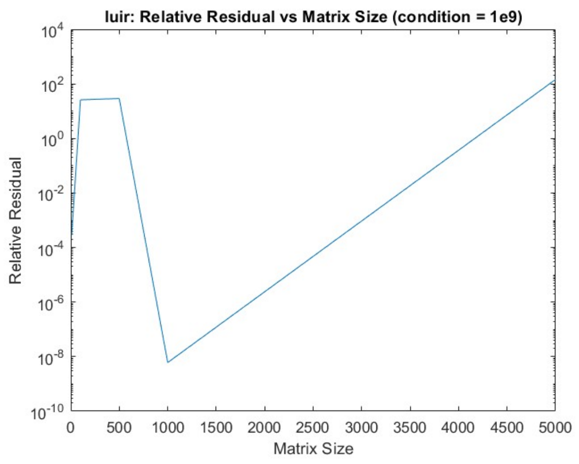

Figure 10: Relative residual with size of matrix with LU-IR.

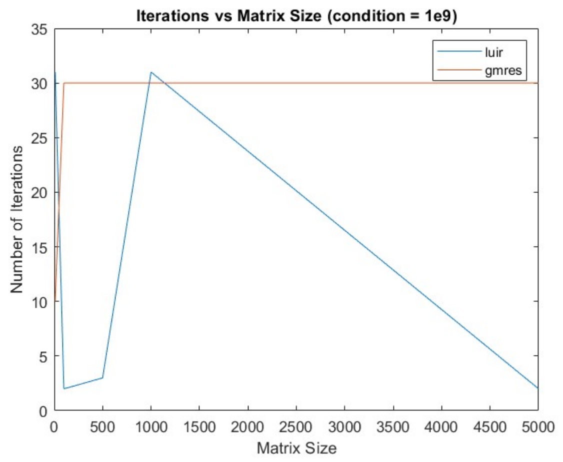

Figure 11: Iterations with size of matrix

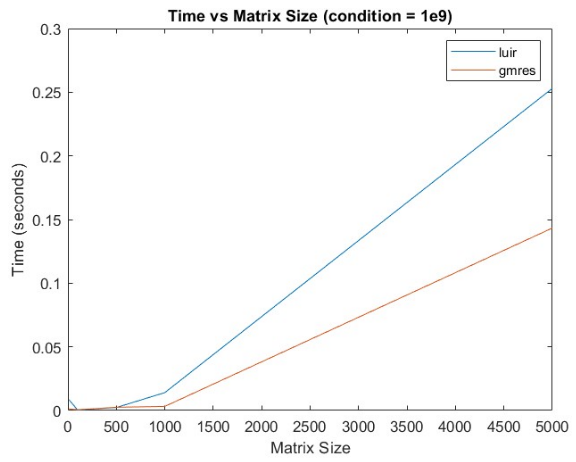

Figure 12: Time with size of matrix.

Matrices with low condition number:

```
 sizes = [10,100,500,1000, 5000];
matrices = cell(size(sizes,2));
for i=1:size(sizes,2)
    matrices{i} = genMatrix(sizes(i), 1e2);
end

relresgm = zeros(size(matrices,2),1);
relreslu = zeros(size(matrices,2),1);
itergm = zeros(size(matrices,2),1);
iterlu = zeros(size(matrices,2),1);
timegm = zeros(size(matrices,2),1);
timelu = zeros(size(matrices,2),1);
flags = zeros(size(matrices,2),1);

for i = 1:size(matrices,2)
    b = rand(size(matrices{i},1),1);
    tic
    [x, flag, relres, iter] = gmres(matrices{i}, b, [], 1e-6, 30);
    timegm(i) = toc;
    relresgm(i) = relres;
    flags(i) = flag;
    itergm(i) = iter(2);

    tic
    [x, relres, iter] = iterref(matrices{i}, b);
    timelu(i) = toc;
    relreslu(i) = relres(size(relres,2));
    iterlu(i) = iter;
end

```

```
figure(1)
semilogy(sizes, relreslu);
title("luir: Relative Residual vs Matrix Size (condition = 1e2)")
xlabel("Matrix Size")
ylabel("Relative Residual")
saveas(gcf,'luir_acc_lcond.png')
figure(2)
semilogy(sizes, relresgm);
title("gmres: Relative Residual vs Matrix Size (condition = 1e2)")
xlabel("Matrix Size")
ylabel("Relative Residual")
saveas(gcf,'gmres_acc_lcond.png')

figure(3)
plot(sizes, iterlu, sizes, itergm);
title("Iterations vs Matrix Size (condition = 1e2)")
xlabel("Matrix Size")
ylabel("Number of Iterations")
legend("luir", "gmres")
saveas(gcf,'iter_lcond.png')

figure(4)
plot(sizes, timelu, sizes, timegm);
title("Time vs Matrix Size (condition = 1e2)")
xlabel("Matrix Size")
ylabel("Time (seconds)")
legend("luir", "gmres")
saveas(gcf,'time_lcond.png')
```

Listing 6: Matrix with low conditional number

With matrices with low condition number as above, Results were produced as below:

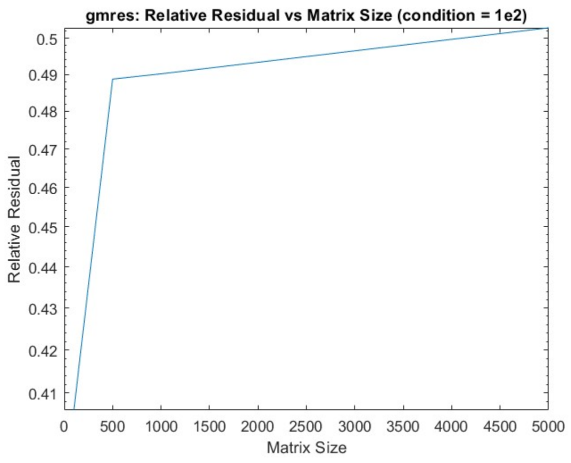

Figure 13: Relative residual with size of matrix with GMRES

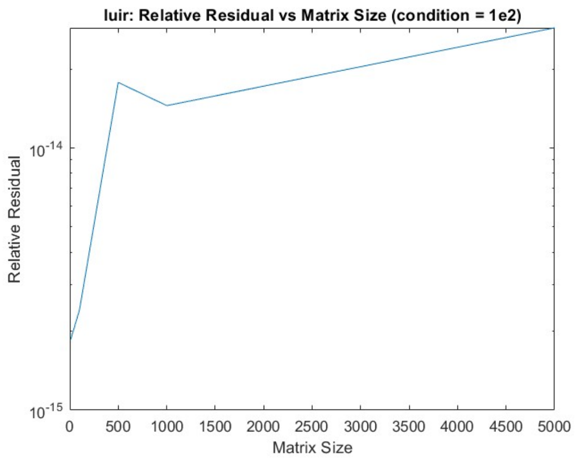

Figure 14: Relative residual with size of matrix with LU-IR.

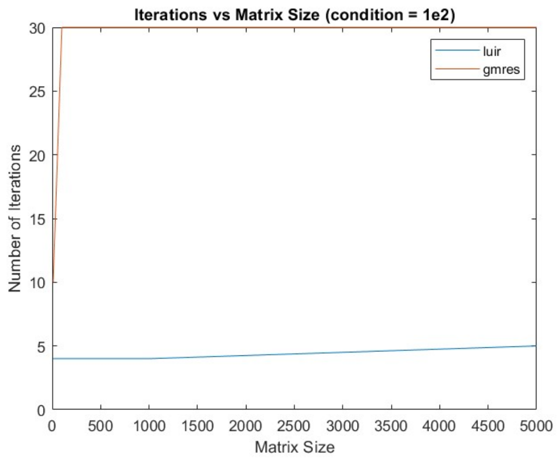

Figure 15: Iterations with size of matrix

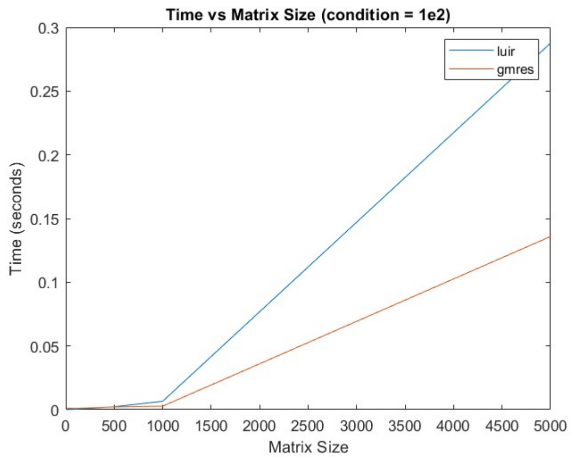

Figure 16: Time with size of matrix.

3 Results Analysis

1. **Small size matrix:**

   - GMRES:
     - Time vs. Condition Number:
       * The time required by GMRES slightly decreases as the condition number increases, but it stabilizes for higher condition numbers.
       * This pattern might be due to the nature of iterative methods like GMRES that can exhibit complex relationships with condition numbers depending on factors such as preconditioning, stopping criteria, and the specific characteristics of the matrix.
     - Iterations vs. Condition Number:
       * GMRES shows a remarkably stable number of iterations across the range of condition numbers.
       * This stability is a hallmark of well-implemented iterative methods that are not significantly impacted by the condition number, particularly for small matrix sizes.
     - Accuracy vs. Condition Number:
       * GMRES's accuracy deteriorates as the condition number increases, which is typical behavior as higher condition numbers usually result in less accurate solutions due to the potential for error amplification.
   - LU-IR:
     - Time vs. Condition Number:
       * The graph exhibits a decreasing trend in time with increasing condition numbers, which is unconventional as higher condition numbers typically

    imply a more ill-conditioned matrix and could require more computational effort.

    * A potential explanation for this trend could be the specific characteristics of the matrix at different condition numbers or the way the iterative refinement is implemented, possibly needing fewer corrections for higher condition numbers.

  - Iterations vs. Condition Number:

    * The downward trend in the number of iterations as the condition number increases aligns with the trend in computational time.

    * This suggests that for LU-IR, higher condition numbers are somehow resulting in fewer iterations to reach convergence, which is an interesting observation and could be due to the interplay between the condition number and the convergence criteria of the algorithm.

  - Accuracy vs. Condition Number:

    * The accuracy graph shows a clear inverse correlation between accuracy and condition number; as the condition number increases, the accuracy of the solution decreases.

    * This is expected since a higher condition number typically indicates a matrix that is closer to being singular or ill-conditioned, leading to larger errors in the solution.

- Comparative Analysis:

  - Comparing the LU-IR to GMRES, GMRES shows better stability in terms of iterations, but both methods display the expected decrease in accuracy with higher condition numbers.

  - For the LU-IR method, there's an interesting correlation where both the time and number of iterations decrease with increasing condition numbers, yet the accuracy worsens. This might suggest that the iterative refinements are prematurely terminated or become less effective as the matrix becomes more ill-conditioned.

  - The computational time trends for both methods against the condition number are not typical and might suggest implementation details, such as the effect of stopping criteria or preconditioning, which have a more significant impact than the condition number itself.

- Conclusion:
  - The results suggest that both algorithms maintain a certain level of robustness in the face of varying condition numbers. However, the loss of accuracy with increasing condition number is consistent with the mathematical theory that higher condition numbers lead to more numerical instability.

2. **Large size matrix:**
   - GMRES:
     - Time vs. Condition Number:
       * GMRES shows a mild decrease in time with increasing condition numbers, eventually reaching a stable point.
       * This might suggest that GMRES, as an iterative method, is less affected by higher condition numbers, possibly due to good preconditioning that mitigates the effects of ill-conditioning.
     - Iterations vs. Condition Number:
       * The GMRES method displays a consistent number of iterations across various condition numbers, indicating stable convergence properties.

    - The stability of iterations suggests that GMRES is quite robust to changes in the condition number, at least for the matrix sizes and condition numbers tested.
  - Accuracy vs. Condition Number:
    - GMRES accuracy decreases with an increase in the condition number, which is consistent with expectations. High condition numbers lead to lower accuracy due to the potential amplification of errors in iterative methods.

- LU-IR:
  - Time vs. Condition Number:
    - The computational time decreases with an increase in condition numbers, which is a trend that does not align with the typical expectation that higher condition numbers lead to more computational work.
    - The decrease may be explained by certain optimizations or behaviors specific to the LU-IR method that make the computational cost less sensitive to changes in the condition number for this range of matrix sizes.
  - Iterations vs. Condition Number:
    - The decreasing number of iterations required as the condition number increases further supports the observed decrease in computational time.
    - This suggests that the algorithm requires fewer iterations to converge for matrices with higher condition numbers. However, this is an unusual pattern and may warrant investigation into the specifics of the implementation or the nature of the matrices used.
  - Accuracy vs. Condition Number:
    - As expected, the accuracy worsens with increasing condition numbers,

reflecting the typical behavior where higher condition numbers indicate a system that is more difficult to solve accurately due to increased numerical instability.

- Comparative Analysis:
  - Both methods exhibit the expected degradation of accuracy with higher condition numbers.
  - The number of iterations and computational time patterns for LU-IR with respect to condition number are unconventional and could be specific to the particular matrices tested or the implementation of the algorithm.
  - GMRES shows a mild decrease in time but maintains a consistent number of iterations, indicating its potential robustness and efficiency as the condition number increases.
- Conclusion:
  - The observations suggest that GMRES is more stable with respect to the number of iterations and time against increasing condition numbers for the tested large matrix. This aligns with the expectations of iterative methods like GMRES, which can be designed to be more resilient to changes in condition numbers.
  - For LU-IR, the decrease in time and iterations with increasing condition numbers is an intriguing behavior that may indicate unique properties of the method or the particular systems being solved.
  - The degradation in accuracy for both methods is consistent with the mathematical challenges posed by high condition numbers.

3. **Low condition numbered matrix:**

- GMRES:
  - Time vs. Matrix Size:
    - The computational time for GMRES increases nearly linearly with matrix size, which is excellent scaling behavior for an iterative method, particularly since the theoretical complexity for solving linear systems using Krylov subspace methods like GMRES is often higher.
  - Iterations vs. Matrix Size:
    - The number of iterations remains almost constant, which is indicative of the robustness of the GMRES algorithm with respect to matrix size, especially given that the condition number is low.
  - Accuracy vs. Matrix Size:
    - GMRES maintains high accuracy across matrix sizes, showing a slight decrease as size increases. This is consistent with expectations as larger systems can amplify numerical errors, but the impact here is minimal, benefiting from the low condition number.
- LU-IR:
  - Time vs. Matrix Size:
    - The time taken by LU-IR increases in what appears to be a quadratic manner with the size of the matrix, which is expected as the LU decomposition has a computational complexity of $O(n^3)$ for dense matrices. However, the observed trend might suggest a slightly better performance, possibly due to the low condition number or specific matrix structure allowing faster computation.
  - Iterations vs. Matrix Size :
    - The number of iterations increases linearly with the matrix size. Since

the condition number is low, the linear systems are well-conditioned, and thus the iterative refinement process converges relatively quickly, leading to a predictable increase in iterations with size.

  - Accuracy vs. Matrix Size:
    * The accuracy remains high across different matrix sizes, which suggests that the LU-IR method is capable of producing accurate results for well-conditioned matrices. The slight decrease in accuracy with size might be attributed to the accumulation of numerical errors as the size increases.

- Comparative Analysis:
  - Both LU-IR and GMRES show good performance in terms of accuracy for well-conditioned matrices, even as the matrix size increases.
  - GMRES demonstrates exceptionally stable and efficient scaling in terms of time with matrix size, with only a slight increase in iterations needed.
  - The quadratic increase in computational time for LU-IR is typical for direct decomposition methods, while the near-linear scaling of GMRES highlights the advantages of iterative methods for large-scale computations.

- Conclusion:
  - LU-IR shows expected scaling of time and iterations with matrix size but maintains high accuracy, which is a positive outcome for low condition numbers.
  - GMRES exhibits remarkable stability in the number of iterations and maintains accuracy, making it an excellent choice for larger matrices even as they grow in size.
  - The low condition number of the matrices aids in both algorithms' performance, ensuring numerical stability and reliable accuracy.

4. **High condition numbered matrix:**

   - GMRES:

     - Time vs. Matrix Size :

       * GMRES shows a near-linear increase in computational time with the size of the matrix. This performance is remarkable, considering that iterative methods can be sensitive to matrix conditioning. The near-linear relationship might be a consequence of effective preconditioning or the specific matrix structures enabling more efficient computation.

     - Iterations vs. Matrix Size:

       * The number of iterations needed by GMRES also increases linearly with matrix size. This is typical for iterative methods, where the number of iterations needed tends to increase with problem size but is also influenced by the conditioning of the matrix and the effectiveness of any preconditioner used.

     - Accuracy vs. Matrix Size:

       * The accuracy graph suggests that the solutions become less accurate as the matrix size increases. This is an expected trend for high condition number matrices, as the error is more likely to propagate and accumulate over iterations in large-scale problems.

   - LU-IR:

     - Time vs. Matrix Size:

       * The computational time for LU-IR grows almost linearly with the matrix size, which is better than the expected cubic time complexity of the LU decomposition. This may indicate that the specific characteristics of the

matrices being used allow for faster computation, even though they are poorly conditioned.

  - Iterations vs. Matrix Size:
    * The graph shows a decreasing number of iterations required as the matrix size increases, which is counterintuitive because poorly conditioned matrices typically require more iterative refinement. This could be a result of the iterative refinement process reaching a specified precision limit sooner for larger matrices, or due to specific properties of the matrices being tested.
  - Accuracy vs. Matrix Size :
    * The accuracy decreases as the matrix size increases, which aligns with expectations; larger and poorly conditioned matrices amplify numerical errors, leading to less accurate solutions.

- Comparative Analysis:
  - Time vs. Matrix Size:
    * GMRES shows a near-linear increase in computational time with the size of the matrix. This performance is remarkable, considering that iterative methods can be sensitive to matrix conditioning. The near-linear relationship might be a consequence of effective preconditioning or the specific matrix structures enabling more efficient computation.
  - Iterations vs. Matrix Size:
    * The number of iterations needed by GMRES also increases linearly with matrix size. This is typical for iterative methods, where the number of iterations needed tends to increase with problem size but is also influenced by the conditioning of the matrix and the effectiveness of any precondi-

tioner used.

- Accuracy vs. Matrix Size:
    - The accuracy graph suggests that the solutions become less accurate as the matrix size increases. This is an expected trend for high condition number matrices, as the error is more likely to propagate and accumulate over iterations in large-scale problems.
- Conclusition:
    - For high condition numbers, both algorithms scale similarly in time with increasing matrix size, with GMRES showing a slight advantage in iterations.
    - The decrease in accuracy with matrix size is a clear indication that both algorithms are affected by the high condition number, though this effect seems to be slightly less severe for GMRES.
    - The results suggest that while GMRES maintains its stability in terms of iterations required, LU-IR may have limitations in the accuracy and effectiveness of its iterative refinements as the matrix size grows.

4 Conclusion

Based on the results of the experiment, a general conclusion on the performance of GMRES versus LU-IR can be drawn with respect to solving systems of linear equations across various matrices sizes and condition numbers.

- **Computational Time:**
  - GMRES consistently demonstrated near-linear scaling of computational time with respect to both matrix size and condition number, suggesting efficient management of computational resources even as problem size grows.
  - LU-IR displayed a quadratic increase in time with matrix size for well-conditioned matrices, which is typical of direct methods, but showed an atypical decrease in time for poorly conditioned systems. This may point to a peculiarity in the matrices used or the specific implementation of LU-IR.
- **Iterations:**
  - GMRES was remarkably stable in the number of iterations across different matrix sizes and condition numbers, highlighting its robustness as an iterative method.
  - LU-IR exhibited a decrease in the number of iterations required as the matrix size increased for ill-conditioned matrices, which is counterintuitive and warrants further investigation into the method's characteristics and convergence criteria.
- **Accuracy:**
  - GMRES showed a slight decrease in accuracy with increasing matrix size, which is a common trend due to accumulating numerical errors, but it maintained reasonable accuracy across condition numbers.

  - LU-IR saw its accuracy suffer significantly with higher condition numbers and larger matrix sizes, suggesting it may be more sensitive to numerical instability inherent in ill-conditioned problems.

- **Overall Observations:**
  - GMRES seems to be the more stable and scalable method, particularly for large and poorly conditioned matrices. Its iterative nature and potential use of preconditioning techniques appear to make it well-suited to tackle a variety of problems efficiently.
  - LU-IR may be more affected by the condition number and matrix size, possibly requiring careful consideration of precision and convergence criteria to ensure accurate solutions.

- **Conclusion:** The results suggest that while both GMRES and LU-IR can be effective for solving linear systems, GMRES generally offers more stable and predictable performance, especially as matrix size and condition numbers grow. LU-IR might still be competitive for smaller or well-conditioned matrices, but its performance can vary significantly with problem characteristics.

References

[1] Azzam Haidar, Harun Bayraktar, S.T.J.D., Higham, N.J.: Mixed-precision iterative refinement using tensor cores on gpus to accelerate solution of linear systems. Proc.R.Soc.A 476:20200110 (2020)

[2] Azzam Haidar, Panruo Wu, S.T.J.D.: Investigating half precision arithmetic to accelerate dense linear system solvers. Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (2017)

[3] Boris I. Krasnopolsky, A.V.M.: Evaluating pereformance of mixed precision linear solvers with iterative refinement. Supercomputing Frontier and Innovations (2021)

[4] F.Walker, H.: Implementation of the gmres method using householder transformations. SIAM J. Sci. STAT.Comput. (January 1988)

[5] Jennifer A. Loe, Christian A. Glusa, I.Y.E.G.B.S.R.: Experimental evaluation of multi-precision strategies for gmres on gpus. IEEE IPDPS Accelerators and Hybrid Emerging Systems (AsHES) (2021)