# LU versus Using the Inverse Matrix

The equation $\mathbf{Ax} = \mathbf{b}$ can be solved using an LU factorization, or using the inverse matrix via the formula $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. This note compares these methods. This is motivated by a posted, but unpublished, paper: How Accurate is inv(A)*b?. In this paper the authors state that: "Many textbooks, including recent and widely-used ones, mislead the reader to think that $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ is less accurate than $\mathbf{x} = \mathbf{A}\backslash\mathbf{b}$, which computes the LU factorization."

The approach here is purely experimental, so there is no mathematical analysis for any differences between these methods. The calculations are done using MATLAB, version R2025a, on a Mac mini (with 64GB of memory and a M4 Pro chip with 14 cores, running OS 15.6).

### Example 1: Random

The matrix and the solution are randomly generated. The MATLAB commands are (the entire code is attached):

```
A=rand(n,n);
x=rand(n,1);
b=A*x;
```

The equation was solved and the error $||\mathbf{x} - \mathbf{x}_c||_\infty$ and computing time (using tic/toc) determined. For each `n` this was done 1000 times and the results averaged. The results are shown in Figure 1.

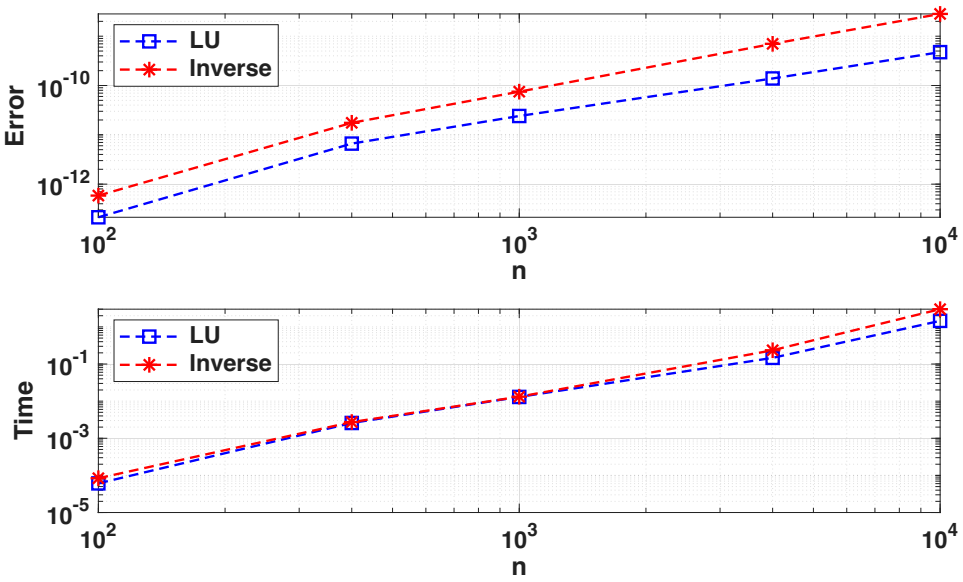


*Figure 1: Comparison for randomly generated matrices and solutions.*

**Example 2: Laplace**

This example reconsiders Example 9.4 in the text. This involves the matrix equation coming from solving Laplace's equation. Now, the matrix is fixed but the solution is randomly generated and the results averaged. The results are shown in Figure 2, when the sparse command is not used, and in Figure 3 when it is used (only the time is shown as the error is unchanged).

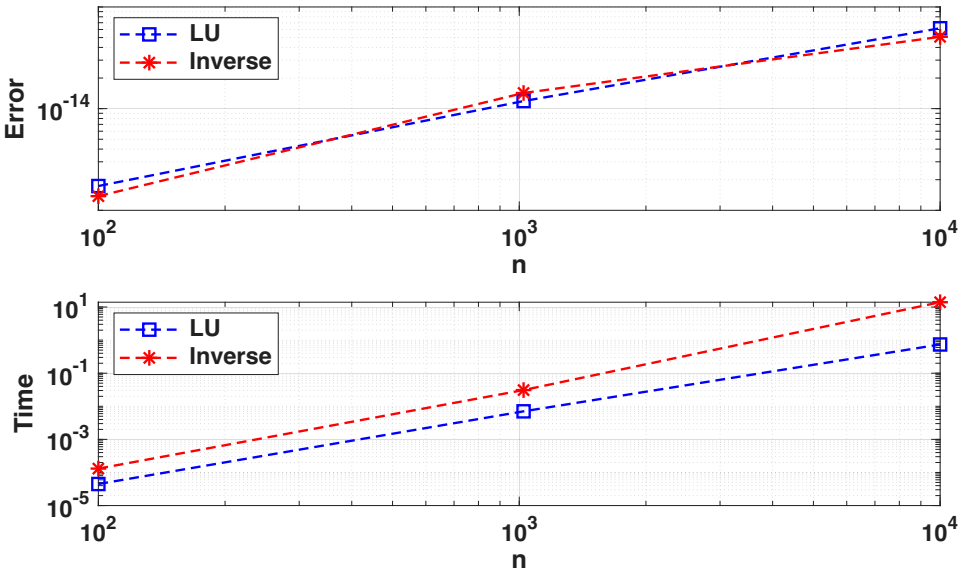


*Figure 2: Comparison when solving Laplace's equation.*

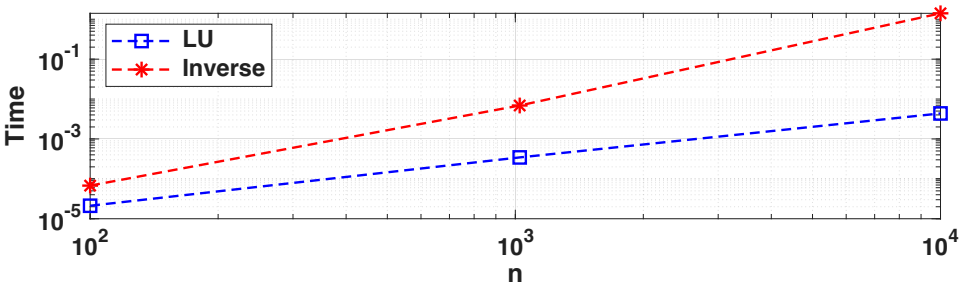


*Figure 3: Comparison in the computing time when solving Laplace's equation when using the sparse command.*

**Observations**

In Example 1 the $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ result is less accurate than using LU, and the difference increases with $n$. It is also slower than LU, for larger $n$, although not by the factor of 3 predicted using a flop count. In Example 2 the two methods are approximately the same in terms of the error, but LU is significantly faster. This is due, in large part, to MATLAB using a Cholesky factorization. This is the point of this example as Cholesky is simply a LU specialized to a symmetric and positive definite matrix. The inverse approach does not take advantage of these two properties.

Based solely on these two examples, LU is the method of choice.

```
function matrixB

% solve matrix equation: random A and x

ns=[100; 400; 1000; 4000; 10000];
N=length(ns);
NN=1000;
error=zeros(N,1);
error2=zeros(N,1);
t=zeros(N,1);
t2=zeros(N,1);

for in=1:N
    n=ns(in);

    for it=1:NN
        A=rand(n,n);
        xe=rand(n,1);
        b=A*xe;
        % solve equation
        % using LU
        tic;
        xc=A\b;
        t(in)=t(in)+toc;
        % using inverse
        tic;
        xc2=inv(A)*b;
        t2(in)=t2(in)+toc;
        error(in)=error(in)+norm(xe-xc,inf);
        error2(in)=error2(in)+norm(xe-xc2,inf);
    end

end

% plot error
figure(1)
clf
% get(gcf)
set(gcf,'Position', [25 930 660 420])
subaxis(2,1,1,1,'MT',0.01,'MB',0.08,'MR',-0.02,'ML',0.05,'P',0.05)
co = [0 0 1;
    0 0.5 0;
    1 0 0;
    0 0.75 0.75;
    0.75 0 0.75;
    0.75 0.75 0;
    0.25 0.25 0.25];
set(groot,'defaultAxesColorOrder',co)

loglog(ns,error/NN,'--sb','LineWidth',1.2,'MarkerSize',8)
hold on
loglog(ns,error2/NN,'--*r','LineWidth',1.2,'MarkerSize',8)
```

```
set(gca,'ytick',[1e-14 1e-12 1e-10 1e-8])
legend({' LU','
Inverse'},'AutoUpdate','off','Location','NorthWest','FontSize',12,'FontWeight
','bold')
xlabel('n')
ylabel('Error')
grid on
box on
set(gca,'FontSize',12,'FontWeight','bold')

subaxis(2,1,1,2)
loglog(ns,t/NN,'--sb','LineWidth',1.2,'MarkerSize',8)
hold on
loglog(ns,t2/NN,'--*r','LineWidth',1.2,'MarkerSize',8)
set(gca,'ytick',[1e-5 1e-3 1e-1 1e1])
legend({' LU','
Inverse'},'AutoUpdate','off','Location','NorthWest','FontSize',12,'FontWeight
','bold')
xlabel('n')
ylabel('Time')
grid on
box on
set(gca,'FontSize',12,'FontWeight','bold')

exportgraphics(gcf,'/Users/mark/Desktop/comparison.pdf')
```

*Published with MATLAB® R2025a*