Computational Time for Finding the Inverse of a Matrix: LU Decomposition vs. Naive Gaussian Elimination

Jamie Trahan, Autar Kaw, Kevin Martin
University of South Florida
United States of America
kaw@eng.usf.edu

Introduction

This worksheet demonstrates the use of *Mathematica* to illustrate the computational time needed to find the inverse of a matrix using two different methods: LU Decomposition and Naive Gaussian Elimination. Although both methods have similarities, this worksheet will prove that one method is computationally more efficient than the other.

Section 1: Background: Inverse of a Matrix

To find the inverse of a $[A]_{nxn}$ matrix, we need to find a matrix $[B]_{nxn}$ such that [A][B]=[I]=[B][A] where $[I]_{nxn}$ is an identity matrix. This implies the j^{th} column $[X]_{nx1}$ of the inverse matrix $[B]_{nxn}$ corresponds to the solution of [A][X]=[C], where $[C]_{nx1}$ is the j^{th} column of the identity matrix.

Section 2: Definitions

The execution time of a program depends on the number of *floating*-point operations (FLOPs) involved. Every computer has a processor speed which can be defined in flops/sec. Knowing the processor speed and how many flops are needed to run a program gives us the computational time required:

Time required (sec) = Number of FLOPs/Processor Speed (FLOP/sec)

A supercomputer may be capable of 50×10^{12} FLOPs per second, while a typical PC may be capable of 10×10^{9} FLOPs per second.

Section 3: Computational methods of solving the equations

The problem of finding the inverse of a *n*x*n* [A] matrix reduces to solving *n* sets of equations with the *n* columns of the identity matrix as the RHS vector. Complete details are given here.
The formulas that define the number of FLOPs required to find the inverse of a matrix using Naïve Gauss Elimination and LU Decomposition are given below.

■ *Inverse using Naïve Gaussian Elimination:*

To find the inverse of a *n*x*n* matrix, one can use Naïve Gaussian Elimination method. For calculations of *n* columns of the inverse of the matrix, the forward elimination and back substitution needs to be done *n* times. Complete details of Naïve Gauss Elimination are given here.

The following formulas define the number of FLOPs for each step of Naïve Gauss method.

Forward Elimination (FENG): The FLOPs used in the forward elimination step of Naïve Gauss for a set of *n* equations is given by the series

$$\sum_{k=1}^{n-1} (n*(n+2)-k*(2*n+2)+k\text{^}2)$$

$$\frac{1}{6}(-1+n)\,n\,(5+2\,n)$$

When expanded, the number of FLOPs used is equal to

$$\text{FENG} = \text{Expand}\left[\sum_{k=1}^{n-1} (n\,(n+2)-k\,(2\,n+2)+k\text{^}2)\right]$$

$$-\frac{5\,n}{6}+\frac{n^2}{2}+\frac{n^3}{3}$$

Back Substitution (BSNG): The FLOPs used in the back substitution step for a set of *n* equations is given by the series

$$\sum_{i=1}^{n} i$$

$$\frac{1}{2} n (1 + n)$$

When expanded, the number of FLOPs is equal to

```
BSNG = Expand[∑_{i=1}^{n} i]
```

$$\frac{n}{2} + \frac{n^2}{2}$$

Total number of FLOPs required to find the inverse of the [A] matrix using Naïve Gaussian Elimination is n*(FE+BS) which is equivalent to:

```
NGFLOP = Expand[n * (FENG + BSNG)]
```

$$-\frac{n^2}{3} + n^3 + \frac{n^4}{3}$$

■ *Inverse using LU Decomposition*

To find the inverse of a *nxn* matrix, one can use LU Decomposition method. For calculations of each column of the inverse of the matrix, the coefficient matrix in the set of equations does not change. So if we use LU Decomposition method, the decomposition needs to be done only once, and the forward substitution and back substitution needs to be done *n* times each. Complete details are explained here.

The following formulas define the number of FLOPs for each step of LU Decomposition.

Forward Elimination (FELU): The FLOPs used in forward elimination to find the [L][U] decomposition is given by the series

```
∑_{k=1}^{n-1} (n (n + 2) - k (2 n + 2) + k^2)
```

$$\frac{1}{6} (-1 + n) n (5 + 2 n)$$

When expanded, the series defines the number of FLOPs used as

```
FELU = Expand[∑_{k=1}^{n-1} (n (n + 2) - k (2 n + 2) + k^2)]
```

$$-\frac{5\,n}{6}+\frac{n^2}{2}+\frac{n^3}{3}$$

Forward Substitution (FSLU): The FLOPs used in forward substitution for a set of *n* equations is given by the series

$$\sum_{i=1}^{n} i$$

$$\frac{1}{2}\,n\,(1+n)$$

When expanded, the FLOPs used is given by the formula

```
FSLU = Expand[∑_{i=1}^{n} i]
```

$$\frac{n}{2}+\frac{n^2}{2}$$

Backward Substitution (BSLU): The FLOPs used in back substitution for a set of *n* equations is given by the series

$$\sum_{i=1}^{n} i$$

$$\frac{1}{2}\,n\,(1+n)$$

Once expanded, the FLOPs required is determined by the following formula:

```
BSLU = Expand[∑_{i=1}^{n} i]
```

$$\frac{n}{2}+\frac{n^2}{2}$$

Total number of FLOPs required to find the inverse of the [A] matrix using LU Decomposition is (FELU + n*(FSLU+-BSLU)) or

```
LUFLOP = Expand[FELU + n (FSLU + BSLU)]
```

$$-\frac{5\,n}{6}+\frac{3\,n^2}{2}+\frac{4\,n^3}{3}$$

Section 4: Example

For a small square matrix, let us say n=10, the number of floating-point operations using Naïve Gaussian Elimination is:

```
NGFLOP /. n → 10
```

```
4300
```

For the same size matrix, the number of FLOPs using LU Decomposition is

```
LUFLOP /. n → 10
```

```
1475
```

For a matrix of this size, Naïve Gaussian method requires nearly 3 (or approximately n/4) times more FLOPs than LU Decomposition method. However, if one were to calculate the FLOPs required for a square matrix with an order of 100, one can see that, although the order of the matrix increases 10 fold, the number of FLOPs for Naïve Gaussian Elimination requires nearly 20 (or approximately *n*/4) times more FLOPs than LU Decomposition.

FLOPs for Naïve Gauss:

```
NGFLOP /. n → 100
```

```
34330000
```

FLOPs for LU Decomposition:

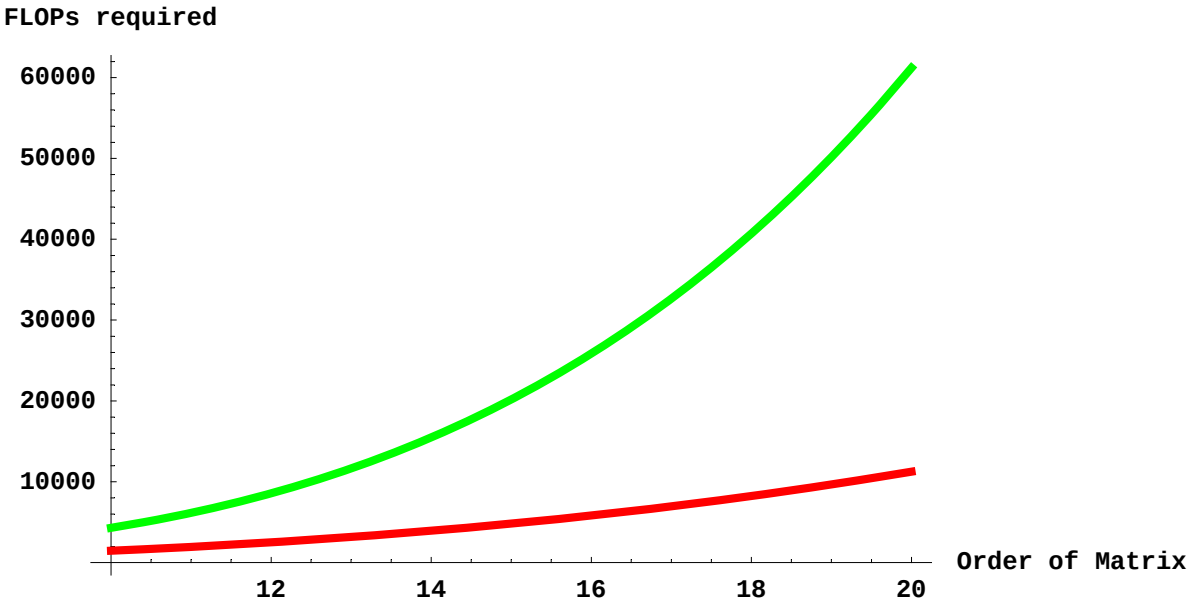
```
LUFLOP /. n → 100
```

```
1348250
```

Section 5: Comparison Plots

Below is a plot that shows the FLOPs required for finding the inverse of a matrix using both Naïve Gauss Elimination (in green) and LU Decomposition (in red).

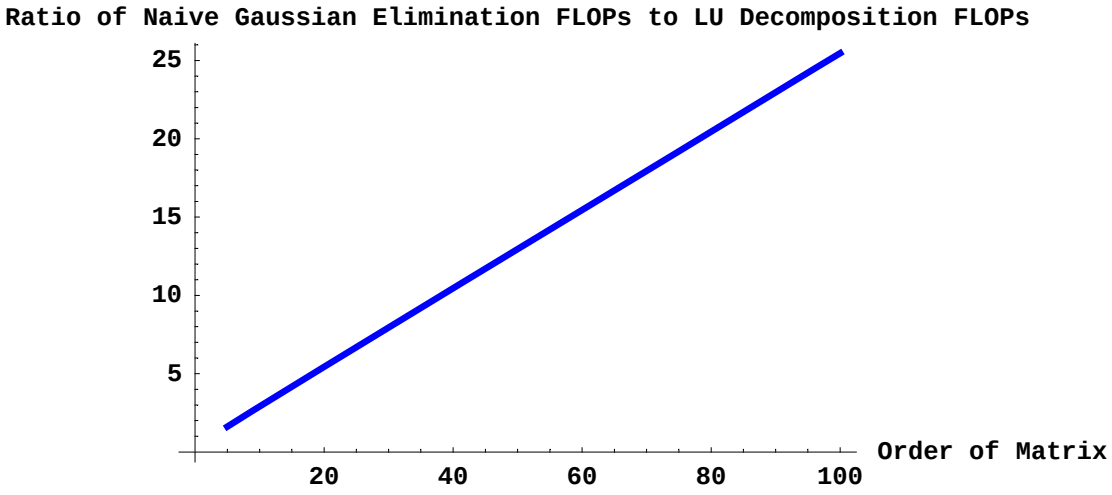
```
Plot[{LUFLOP, NGFLOP}, {n, 10, 20}, PlotStyle →
  {{Thickness[0.010], RGBColor[1, 0, 0]}, {Thickness[0.010], RGBColor[0, 1, 0]}},
 AxesLabel → {"Order of Matrix", "FLOPs required"},
 TextStyle → {FontWeight → "Bold", FontSize → 12}]
```




```
- Graphics -
```

The graph that follows plots the ratio of Naïve Gauss Elimination FLOPs to LU Decomposition FLOPs as a function of the order of the matrix *n*.

```
plot2 = Plot[Evaluate[NGFLOP / LUFLOP], {n, 5, 100},
  PlotStyle → {Thickness[0.010], RGBColor[0, 0, 1]}, AxesLabel → {"Order of Matrix", ""},
  PlotLabel → "Ratio of Naive Gaussian Elimination FLOPs to LU Decomposition FLOPs",
  TextStyle → {FontWeight -> "Bold", FontSize → 12}]
```




```
- Graphics -
```

Conclusion

Using *Mathematica*, we are able to show the computational efficiency of finding the inverse of a square matrix using LU Decomposition method over Naïve Gaussian Elimination method. The LU Decomposition method is $n/4$ times more efficient in finding the inverse than Naïve Gaussian Elimination method.

Question 1: Compare the time in seconds between the two methods to find the inverse of a 10000x10000 matrix on a typical PC with capability of 10×10^9 FLOPs per second.
Question 2: Compare the time in seconds between the two methods to find the inverse of a 1000x1000 matrix on a typical supercomputer with capability of 50×10^{12} FLOPs per second.

References

[1] Autar Kaw, *Holistic Numerical Methods Institute, http://numericalmethods.eng.usf.edu/mws, See*
What is the definition of the inverse of a matrix?
Naive Gaussian Elimination Method
LU Decomposition Method

[2] Chapra & Canale, *Numerical Methods for Engineers*, 4th ed. McGraw-Hill, 2002