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

© 2006 Kevin Martin, Autar Kaw, Jamie Trahan

University of South Florida

United States of America

kaw@eng.usf.edu

Introduction

This worksheet demonstrates the use of Maple to illustrate the computational time needed to find the inverse of a matrix using two different methods: LU Decomposition and Naïve 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

How do I find the 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 that the j^{th} column $[X]_{n\mathrm{x}1}$ of the inverse matrix $[B]_{nxn}$ corresponds to the solution of [A][X] = [C], where $[C]_{n\mathrm{x}1}$ 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 (or *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 (FLOPs/sec)

A supercomputer may be capable of $50\mathrm{x}10^{12}$ (50 trillion) FLOPs per second, while a typical PC may be capable of $10\mathrm{x}10^{9}$ (10 billion) FLOPs per second.

Section 3: Computational methods of solving the equations

The problem of finding the inverse of a *nxn* [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 Gaussian Elimination and LU Decomposition are given below.

3.1 Inverse using Naïve Gaussian Elimination

To find the inverse of a *nxn* matrix, one can use the Naïve Gaussian Elimination method. For calculations of each column 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.

Forward Elimination (FE): The FLOPs used in forward elimination for a set of *n* equations is given by the series (*Reference 2*)

$Sum\left(n\cdot(n+2)-k\cdot(2\cdot n+2)+k^2, k=1..(n-1)\right)$

$$\sum_{k=1}^{n-1}\left(n\,(n+2)-k\,(2\,n+2)+k^2\right) \qquad (4.1.1)$$

```
> FE_NG:=sum('n*(n+2)-k*(2*n+2)+k^2', 'k'=1..n-1);
```

$$FE\_NG := \frac{1}{3}\,n^3+\frac{1}{2}\,n^2-\frac{5}{6}\,n \qquad (4.1.2)$$

Back Substitution (BS): The FLOPs used in back substitution for a set of *n* equations is given by the series (*Reference 2*)

$Sum(i, i=1..n)$

$$\sum_{i=1}^{n} i \qquad (4.1.3)$$

```
> BS_NG:=expand(sum('i','i'=1..n));
```

$$BS\_NG := \frac{1}{2}\,n^2+\frac{1}{2}\,n \qquad (4.1.4)$$

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

```
> NG_FLOP:=expand(n*(FE_NG+BS_NG));
```

$$NG\_FLOP := \frac{1}{3}\,n^4+n^3-\frac{1}{3}\,n^2 \qquad (4.1.5)$$

3.2 Inverse using LU Decomposition

To find the inverse of a *nxn* matrix, one can use the 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.

Forward Elimination (FE): The FLOPs used in forward elimination to find the [L][U] decomposition is (*Reference 2*)

$Sum\left(n \cdot (n+2) - k \cdot (2 \cdot n+2) + k^2, k = 1..(n-1)\right)$

$$\sum_{k=1}^{n-1} \left(n\,(n+2) - k\,(2\,n+2) + k^2\right) \qquad (4.2.1)$$

```
> FE_LU:=sum('n*(n+2)-k*(2*n+2)+k^2', 'k'=1..n-1);
```

$$FE\_LU := \frac{1}{3}\,n^3 + \frac{1}{2}\,n^2 - \frac{5}{6}\,n \qquad (4.2.2)$$

Forward Substitution (FS): The FLOPs used in forward substitution for a set of *n* equations is given by the series (*Reference 2*)

$Sum(i, i = 1..n)$

$$\sum_{i=1}^{n} i \qquad (4.2.3)$$

```
> FS_LU:=expand(sum('i','i'=1..n));
```

$$FS\_LU := \frac{1}{2}\,n^2 + \frac{1}{2}\,n \qquad (4.2.4)$$

Backward Substitution (BS): The FLOPs used in back substitution for a set of *n* equations is given by the series (*Reference 2*)
$Sum(i, i = 1..n)$

$$\sum_{i=1}^{n} i \qquad (4.2.5)$$

```
> BS_LU:=expand(sum('i','i'=1..n));
```

$$BS\_LU := \frac{1}{2}\,n^2 + \frac{1}{2}\,n \qquad (4.2.6)$$

The number of FLOPs required to find the inverse of the [A] matrix using LU Decomposition is:
(FE_LU+n*(FS_LU+BS_LU)

```
> LU_FLOP:=expand(FE_LU+n*(FS_LU+BS_LU));
```

(4.2.7)

$$LU\_FLOP := \frac{4}{3}n^3 + \frac{3}{2}n^2 - \frac{5}{6}n \quad (4.2.7)$$

3.3 Example

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

```
> subs(n=10,NG_FLOP);
                              4300                              (4.3.1)
```

For the same size matrix, the number of floating point operations using LU Decomposition is:

```
> subs(n=10,LU_FLOP);
                              1475                              (4.3.2)
```

For a matrix of this size, Naïve Gaussian method requires nearly 3 (or approximately $n/4 = 12/4 = 3$) times more FLOPs than LU Decomposition method.
However, 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 25 (or approximately $n/4 = 100/4 = 25$) times more FLOPs than LU Decomposition method:

FLOPs for Naïve Gauss given n=100:

```
> subs(n=100,NG_FLOP);
                            34330000                            (4.3.3)
```

FLOPs for LU Decomposition given n=100:

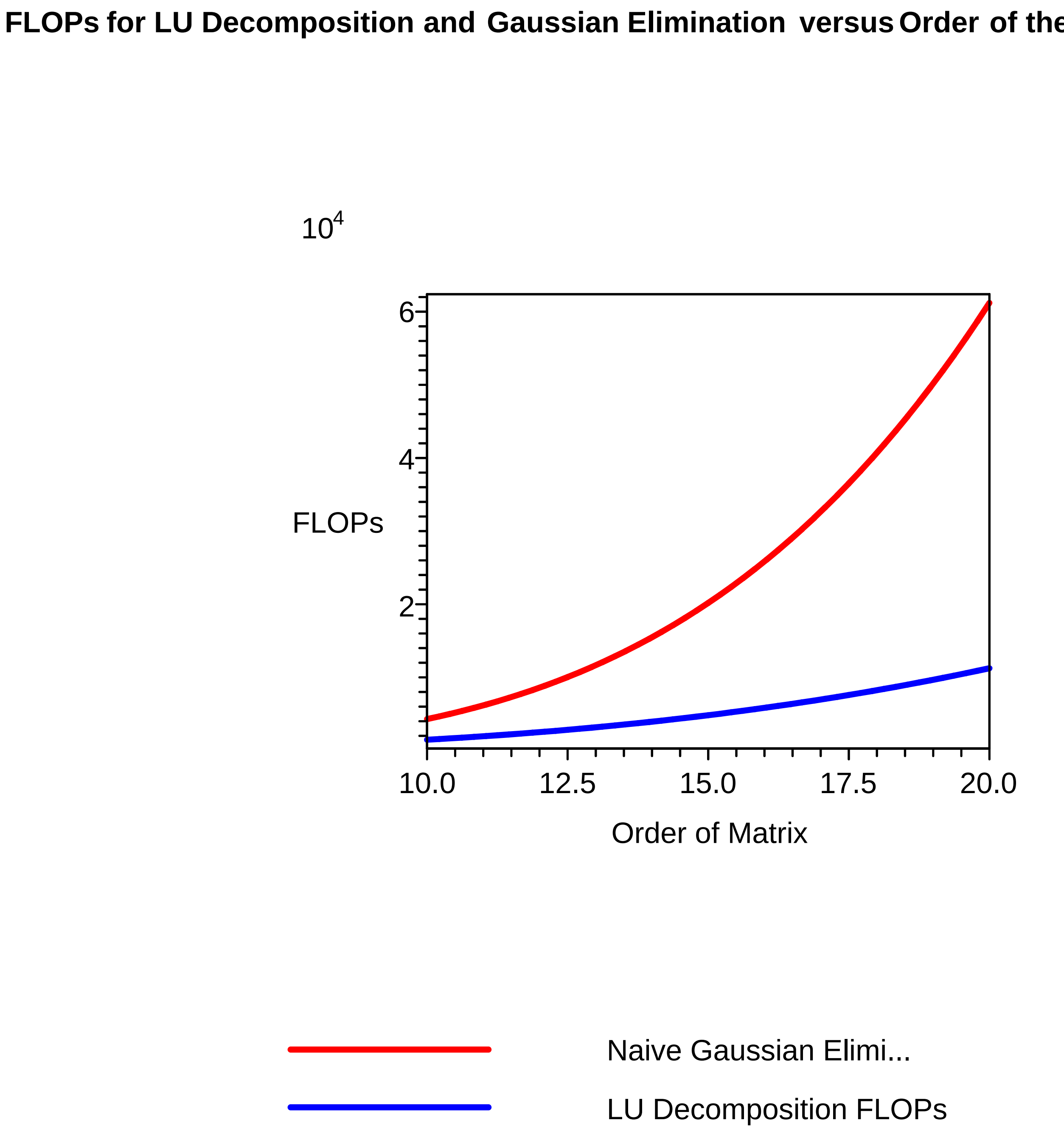
```
> subs(n=100,LU_FLOP);
                            1348250                             (4.3.4)
```

The plots that follow demonstrate the efficiency of using LU Decomposition over Naïve Gaussian Elimination.

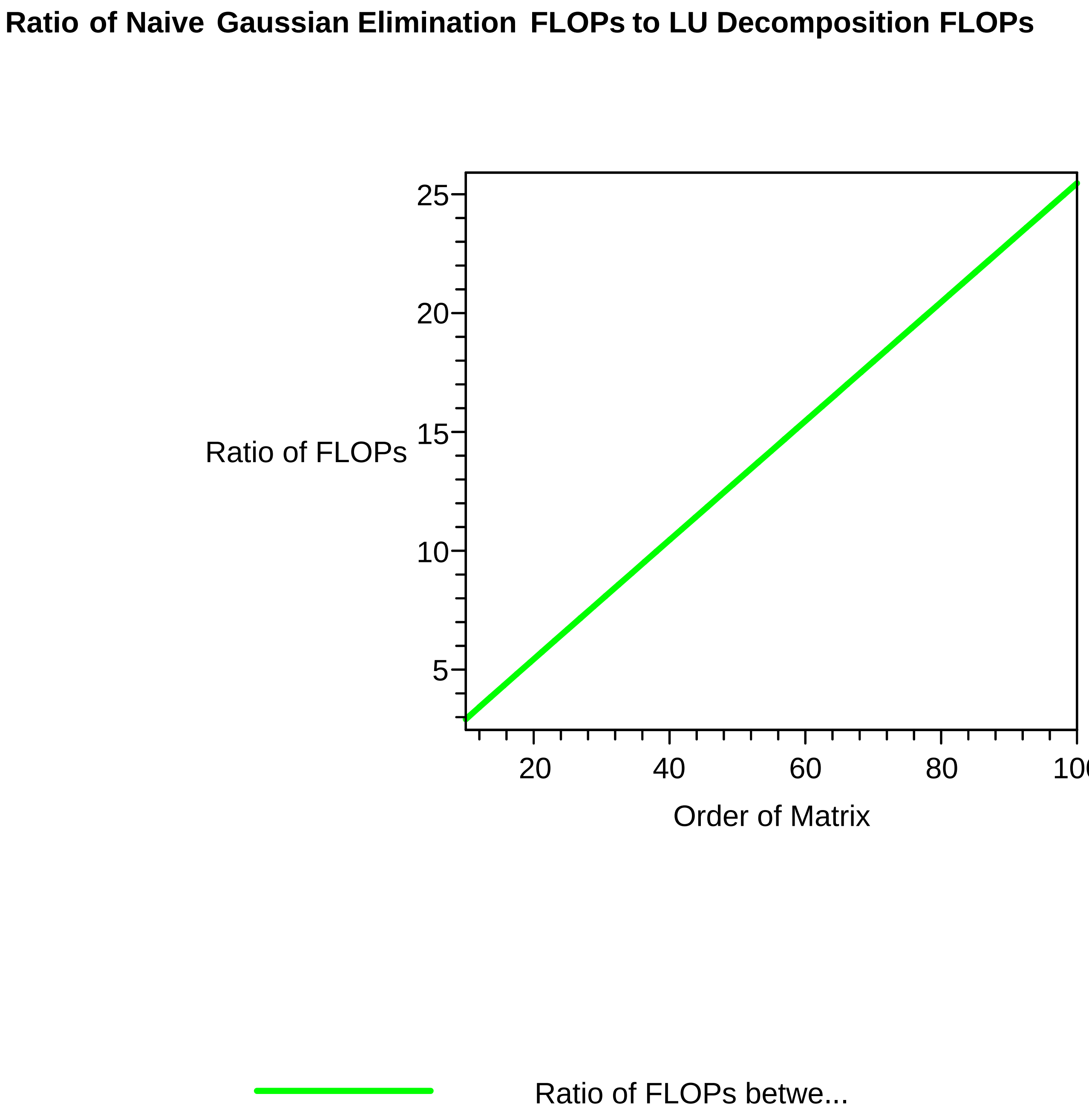
Section 4: Comparison Plots

Below is a plot that shows the FLOPs required for finding the inverse of a matrix using the two methods, as well as a plot for the ratio between the FLOPs required in the two methods.

```
> plot([NG_FLOP,LU_FLOP],n=10..20,color=[red,blue],title=("FLOPs
  for LU Decomposition and Gaussian Elimination versus Order of
  the Matrix"),titlefont=[HELVETICA,BOLD],axes=BOXED,labels=
  ["Order of Matrix","FLOPs"],legend=["Naïve Gaussian Elimination
  FLOPS","LU Decomposition FLOPs"],thickness=2);
```



```
> plot([NG_FLOP/LU_FLOP],n=10..100,color=[green],title=("Ratio of
  Naïve Gaussian Elimination FLOPs to LU Decomposition FLOPs"),
  titlefont=[HELVETICA,BOLD],axes=BOXED,labels=["Order of Matrix",
  "Ratio of FLOPs"],legend=["Ratio of FLOPs between Naïve Gauss
  and LU"],thickness=2);
```



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?
LU Decomposition Method
Naïve Gaussian Elimination Method
[2] Chapra, S.C. and Canale, R.F., "Numerical Methods for Engineers", 4th edition, McGraw-Hill, New York, 2002, pp. 242-243.

Conclusion

Using Maple, 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. 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 between the two methods to find the inverse of a 1000x1000 matrix on a typical supercomputer with the capability of 50×10^{12}
FLOPs per second.