Comparative Analysis of Linear Solvers Using LU Factorization vs. Laplace Expansion

This report examines the performance of linear solvers on equations of varying dimensions and analyses how different algorithms cause a significant performance difference.

1. Introduction

A system of linear equations is defined as:

$$A \cdot x = b$$

Where A is a non-singular[1] matrix of dimensions $m \times n$, x is a vector of $n\times1$ unknowns and b is a $m\times1$ vector. If $m < n$ there are more unknowns than equations so a solution does not exist. Conversely if $n > m$ then the system may be overdetermined. This report considers a non-singular square matrix A where $m = n$ so a unique solution exists.

2. Approaches to solution

a. Laplace Expansion

The "textbook" approach to solving is to directly compute the inverse of A to find x:

$$x = A^{-1} \cdot b$$

Where A^{-1} is obtained by scaling the transpose of the cofactor matrix by the inverse of the determinant. For a n-dimensional matrix with elements a_{ij} (row, column), the cofactor matrix is defined by:

$$C = \begin{bmatrix} (-1)^2 M_{11} & \cdots & (-1)^{1+n} M_{1n} \\ \vdots & (-1)^{i+j} M_{ij} & \vdots \\ (-1)^{n+1} M_{n1} & \cdots & (-1)^{n+n} M_{nn} \end{bmatrix}$$

Where M_{ij} is the determinant of the minor matrix formed by removing the ith row and jth column from A. The determinants are calculated by Laplace Expansion. For example, using Laplace Expansion, the determinant of A will be:

$$\det(A) = \sum (-1)^{i+j} a_{ij} \times M_{ij}$$

Summed along any single row or column. As is evident from the formula, the determinant is a recursive operation on submatrices of A. Then the overall complexity of computing cofactors, and hence the inverse and x (since cofactors are the most expensive computation) becomes:

$$Complexity = n^2\ cofactors \times (n-1)!\ complexity/determinant$$
$$Complexity = n \times n! = O(n!)$$

This means that the computational cost of the inverse scales poorly with matrix size.

[1] Determinant(A) $\neq 0$

b. LU Factorization

LU factorization, and other factorization approaches, seek to bypass the cost of trivially computing the inverse. First, A is factorized as:

$$A = L \cdot U$$

Where L and U are lower and upper triangular matrices of the form:

$$L = \begin{bmatrix} l_{11} & 0 & 0 \\ \vdots & \ddots & 0 \\ l_{n1} & \cdots & l_{nn} \end{bmatrix}, U = \begin{bmatrix} 1 & \cdots & u_{1n} \\ 0 & 1 & \vdots \\ 0 & 0 & 1 \end{bmatrix}$$

Then the product LU looks like:

$$LU = \begin{bmatrix} l_{11} & l_{11}u_{12} & \cdots & l_{11}u_{1n} \\ l_{21} & l_{21}u_{12} + l_{22} & \cdots & \vdots \\ \vdots & \vdots & \ddots & l_{n-1,1}u_{1n} + l_{n-1,2}u_{2n} + \cdots l_{n-1,n-1}u_{n-1,n} \\ l_{n1} & l_{n1}u_{12} + l_{n2} & \cdots & l_{n1}u_{1n} + l_{n2}u_{2n} + \cdots l_{nn} \end{bmatrix} = A$$

An elementwise comparison with A leads to n^2 equations with at most n terms which can be successively solved. Thus the overall complexity of LU Factorization is $O(n^3)$. Given L and U:

$$A \cdot x = b \rightarrow L \cdot U \cdot x = b$$

Let $Ux = d$, then:

$$L \cdot d = b$$

Which can be solved for d using forward substitution ($O(n^2)$). Finally, $Ux = d$ can be solved for x using backwards substitution ($O(n^2)$). Thus:

$$Complexity = O(n^3) + O(n^2) = O(n^3)$$

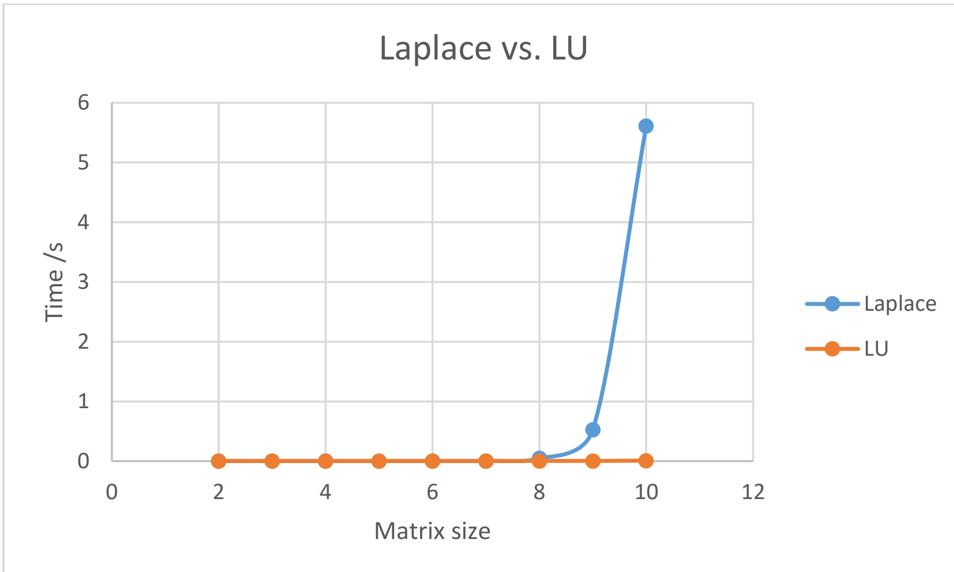
3. Testing[2]

Both approaches were tested on linear systems with dimensions from 2 to 10. The matrices were randomly generated. The Laplace Expansion method was coded from scratch whereas LU factorization was implemented using PETSc by setting up a Krylov solver with default options. The following results were obtained (incremental changes not registered at current level of precision):

n	LU	Laplace
2	0.004	0
3	0.004	0
4	0.004	0
5	0.004	0
6	0.004	0
7	0.004	0.004
8	0.004	0.048

[2] Source code: https://github.com/hazrmard/petsc-learn

9	0.004	0.524
10	0.008	5.612



4. Conclusion

It is evident that the $O(n!)$ complexity of using Laplace Expansion quickly becomes too costly compared to the relatively tame rise in runtime of LU factorization ($O(n^3)$). Other factorization methods to solve linear systems are also based on circumventing the cost of computing determinant using Laplace Expansion.