Draft version October 3, 2018
Preprint typeset using LaTeX style emulateapj v. 12/16/11

COMPARING EFFICIENCY OF DIFFERENT ALGORITHMS FOR SOLVING SETS OF LINEAR EQUATIONS

Markus Bjørklund

Draft version October 3, 2018

ABSTRACT

In this letter I analyze and compare the efficiency of a general, tailored and a LU-decomposition algorithm to solve a set of linear equations. The problem I will focus on is a discretized approximation to the one dimensional Poisson equation. The results show that, for n equations, we get $8n-7$ floating point operations (FLOPS) for the general algorithm, $4n-3$ FLOPS for the tailored algorithm and $\frac{2}{3}n^3$ FLOPS for the LU-decomposition method. I found that the optimal step length is $h = 10^{-7}$. The results also show that the LU decomposition can not handle cases where the matrix is bigger or equal to $n = 10^5$, whilst the other algorithms handle this very well.

Subject headings: Algorithm efficiency — Linear algebra: simulation — methods: computational

1. INTRODUCTION

Solving differential equations is an integral part of most research today. For most differential equations there exists no analytic solution, so we have to use a computational approach. Generally, the more realistic the model is, the more complex it is, and therefore more computationally demanding. There would be a lot to gain to increase efficiency on the algorithms used to solve such equations.

In this letter I will adopt a numerical approach to solving the one dimensional Poisson equation, which shows up in numerous areas of science.

2. METHOD

This section will describe the methods we utilized to acquire our results.

2.1. *Unitary transformations*

Unitary or orthogonal transformations have the property of conserving the dot product and orthogonality. We will show this for orthogonal matrices, but the argument extends to unitary transformations in the complex domain.

Suppose we have two vectors $\mathbf{v}_i$ and $\mathbf{v}_j$. Now, if we consider a orthogonal matrix $\mathbf{U}$, it has the property that $\mathbf{U}^T = \mathbf{U}^{-1}$ such that $\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}$. Applying the transformation, we get new vectors $\mathbf{w}_i = \mathbf{U}\mathbf{v}_i$ and $\mathbf{w}_j = \mathbf{U}\mathbf{v}_j$. Now let's consider the dot product between these new vectors:

$$\begin{aligned}\mathbf{w}_i \cdot \mathbf{w}_j &= (\mathbf{U}\mathbf{v}_i) \cdot (\mathbf{U}\mathbf{v}_j) \\ &= (\mathbf{U}\mathbf{v}_i)^T (\mathbf{U}\mathbf{v}_j) \\ &= \mathbf{U}^T\mathbf{v}_i^T\mathbf{U}\mathbf{v}_j \\ &= \mathbf{v}_i^T\mathbf{v}_j\mathbf{U}^T\mathbf{U} \\ &= \mathbf{v}_i^T\mathbf{v}_j\mathbf{I},\end{aligned}$$

which finally gives us

$$\mathbf{w}_i \cdot \mathbf{w}_j = \mathbf{v}_i \cdot \mathbf{v}_j. \tag{1}$$

which shows the dot product is conserved. If we also assume that $\mathbf{v}_i$ and $\mathbf{v}_j$ is part of an orthogonal basis, such that $\mathbf{v}_i^T \cdot \mathbf{v}_j = \delta_{ij}$, then we from equation 1 that $\mathbf{w}_i \cdot \mathbf{w}_j = \delta_{ij}$ is also true.

3. RESULTS

4. CONCLUSIONS

The results in this letter shows that when faced with a set of linear equations, one has a lot to gain to tailor your algorithm to the special case you are dealing with. Saving CPU time by a factor of 2 is substantial, and comparing to the LU-decomposition, for which the CPU time differs by a factor of n^2, tailoring the algorithm is absolutely essential.

We have also seen that for some cases, the problem is not even solvable by the standard LU-decomposition. For matrices with $n = 10^5$ or above, we did not have enough memory to perform the algorithm. However, the tailored methods work just fine, since we do not have to construct a whole matrix.

4.1. *Further research*

In this letter we have focused on the Poisson equation, with a specific second order approximation. However, similar analysis can be done for several other types of equations, or different approximations. There would be a lot to gain by gaining as efficient methods as possible for each specific case, so that bigger and more complex systems can be simulated and analyzed.

markus.bjorklund@astro.uio.no

REFERENCES

Hjorth-Jensen, Morten Sep 6 2018, "Computational Physics Lectures: Linear Algebra" methods, http://compphysics.github.io/ComputationalPhysics/doc/pub/linalg/pdf/linalg-beamer.pdf

5. APPENDIX

All source codes and figures used can be found at the github repository: https://github.com/marbjo/FYS4150

[1] Institute of Theoretical Astrophysics, University of Oslo, P.O. Box 1029 Blindern, N-0315 Oslo, Norway