FYS 4150 - Computational Physics
Project 1: Solving Poisson's equation in one dimension

Maren Rasmussen   Markus Leira Asprusten   Metin San

4. September 2018

Abstract

1. Introduction

2. Theoretical Background

3. Algorithm & Implementation

3.2 **LU-decomposition**

Solving the linear algebra problem with an LU decomposition is relatively simple. By decomposing the matrix $\hat{\mathbf{A}}$ into the lower triangular matrix $\hat{\mathbf{L}}$ and the upper triangular matrix $\hat{\mathbf{U}}$, where all the diagonal elements in $\hat{\mathbf{L}}$ is 1, in such a way that $\hat{\mathbf{A}} = \hat{\mathbf{L}}\hat{\mathbf{U}}$. We can then rewrite the linear algebra problem into

$$\begin{aligned}
\hat{\mathbf{A}}\hat{\mathbf{u}} &= \hat{\mathbf{f}} \\
\hat{\mathbf{L}}\hat{\mathbf{U}}\hat{\mathbf{u}} &= \hat{\mathbf{f}} \\
\hat{\mathbf{U}}\hat{\mathbf{u}} &= \hat{\mathbf{L}}^{-1}\hat{\mathbf{f}} = \hat{\mathbf{y}} \\
\implies \hat{\mathbf{L}}\hat{\mathbf{y}} &= \hat{\mathbf{f}}\,,\ \hat{\mathbf{U}}\hat{\mathbf{u}} = \hat{\mathbf{y}}.
\end{aligned} \tag{1}$$

The problem is then to solve two equations, firstly for $\hat{\mathbf{y}}$ and lastly for $\hat{\mathbf{u}}$. Suppose the matrices have dimension $(n \times n)$. The solution can then be found by iterating n-times for each equation, to a total of $2n$ iterations. The Armadillo library solves this problem simply with the `solve`-function.

4. Results

4.5 **LU-decomposition**

As described is section *3.2*, the `solve`-function is implemented in `problem_e.cpp`,

n	General Algorithm	Special Algorithm	LU-Decomposition
10^1	$1.2 \cdot 10^{-6}$ s	$1.2 \cdot 10^{-6}$ s	$4.4 \cdot 10^{-5}$ s
10^2	$3.1 \cdot 10^{-6}$ s	$2.9 \cdot 10^{-6}$ s	$8.8 \cdot 10^{-5}$ s
10^3	$2.2 \cdot 10^{-5}$ s	$2.2 \cdot 10^{-5}$ s	$5.2 \cdot 10^{-3}$ s
10^4	$2.2 \cdot 10^{-4}$ s	$2.3 \cdot 10^{-4}$ s	$5.1 \cdot 10^{-1}$ s
10^5	$2.2 \cdot 10^{-3}$ s	$2.2 \cdot 10^{-3}$ s	N/A
10^6	$2.1 \cdot 10^{-2}$ s	$2.1 \cdot 10^{-2}$ s	N/A

Table 1: Average time usage for 500 executions of the different algorithms, tested on Lenovo laptop running linux.

with the number of columns and rows in the square matrix, n, given as a command line input. The run times for this code can be found in table 1. Generally speaking, we see clearly that the LU-decomposition has a longer computational time than the other algorithms to get the same results. Note also that the LU-solver would not run for $n = 10^5$, due to memory allocation error.

Discussion

The relative error in figure **??** is decreasing steadily to about $n = 10^6$, after which it starts increasing. In other words, the error in the calculations is at its lowest at $n = 10^6$. The reason for the increase in error with n larger than this, might seem difficult to explain with an analytical mindset, but there is a lower limit to the precision of a number that a computer can represent. This means that that a larger n will actually make the computations more imprecise as the differences calculated becomes smaller than the machine precision.

The difference in run time between the specialized algorithm discussed in section 4.2 and the general algorithm discussed in section 4.1 seems to depend heavily on the computer doing the operations. As seen in table 1, the results from the Lenovo laptop has a minimal difference in rum time between the general and specialized algorithm, and the small differences there is, might be explained by random chance. The reason for this result is difficult to tell. As seen in table [METINS/MARENS RESULTATER], the results differ when running on another machine. These values are more as expected as the run times for the general algorithm is slower than for the specialized algorithm. The reason this is (and should be) the case, is that the specialized algorithm uses less FLOPS per iteration than the general algorithm.

The results seen in section 4.5 and table 1 shows that the LU-decomposition is noticeably slower than the two other algorithms discussed here. This can be explained by the number of elements that are needed to compute the LU-decomposition. The more specialized only need to to operations with 3 vector of length n, while the solution with LU-decompositions needs to do operations on matrices with dimensions $n \times n$. The problem with this can be seen in table 1, where already at $n = 10^5$, the computer in use does not have enough memory to store all the numbers needed. On the other hand of this problem, the other algorithms exclusively solve a banded matrix, while the LU-decomposition can solve a more general linear algebra problem. So each method has its advantages and drawbacks.