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

Maren Rasmussen Markus Leira Asprusten Metin San

4. September 2018

Abstract

This project involves solving the one-dimensional Possion equation with dirichlet boundary conditions using two different algorithms. The first method is the tridiagonal matrix algorithm while the second is the LU decomposition. The conclusion of the project is that a specialized version of the tridiagonal algorithm is much faster.

1. Introduction

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3. Algorithm

In order to solve Poisson's equation we need to be able to solve the discretized set of equations involving the tridiagonal matrix $\hat{\mathrm{A}}$. We will tackle this problem through the implementation of two algorithms. The first is the Tridiagonal matrix algorithm, also known as the Thomas algorithm. The second is the LU-decomposition algorithm.

3.1. **Tridiagonal Matrix Algorithm.**

This algorithm is a simplified form of Gaussian elimination which can be used to solve tridiagonal systems of equations. A tridiagonal system of n unknowns can be represented as

$$a_i v_{i-1} + b_i v_i + c_i v_{i+1} = b_i, \tag{1}$$

where $a_1 = c_1 = 0$. Or in matrix representation as $\hat{\mathrm{A}}\mathbf{v} = \mathbf{b}$. Written out in the 4×4 case, this becomes

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 \\ a_1 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}. \tag{2}$$

The algorithm starts off by reducing the tridiagonal matrix $\hat{\mathrm{A}}$ to an upper tridiagonal matrix. This is achieved through Gaussian elimination.