

Project 1

Anna Stray Rongve Amund Midtgard Raniseth
Knut Magnus Aasrud

Mandag 9 September 2019

Abstract

Summary of project.

The abstract gives the reader a quick overview of what has been done and the most important results. Try to be to the point and state your main findings.

In project 1 a) we solved a one-dimensional Poisson equation with Dirichlet boundary condition by rewriting it as a set of linear equations, $\mathbf{A}\mathbf{v}=\mathbf{d}$.

In Project 1 b) we solved the linear set of equations $\mathbf{A}\mathbf{v}=\mathbf{d}$, where \mathbf{A} is a tridiagonal matrix, which we expressed as three vectors, a, b, c . To solve our equations we used Gaussian elimination as well as forward and backward substitution.

In Project 1 c) we used our general algorithm to make a special, where the matrix elements along the diagonal were identical

Introduction

The purpose of this project is to implement a numerically effective solution of the one-dimensional Poisson equation

$$-u''(x) = f(x)$$

and to implement this in a programming language of choice (Python, in our case). This will be done using three different approaches - the general Thomas algorithm, a specialized Thomas algorithm and an LU-decomposition - the speed of which is compared.

Theory and technicalities

Conclusion and perspectives

Project 1 a)

We have the discretized version of u, v , with the boundary conditions $v_0 = v_n = 0$:

For $i = 1$

$$-\frac{v_2 + v_0 - 2v_1}{h^2} = f_1$$

For $i = 2$

$$-\frac{v_3 + v_1 - 2v_2}{h^2} = f_2$$

For $i = n - 1$

$$-\frac{v_n + v_{n-2} - 2v_{n-1}}{h^2} = f_{n-1}$$

Multiplying both sides by h^2 gives

$$-v_2 + 2v_1 - v_0 = h^2 \cdot f_1$$

$$-v_3 + 2v_2 - v_1 = h^2 \cdot f_2$$

$$-v_n + 2v_{n-1} - v_{n-2} = h^2 \cdot f_{n-1}$$

Which you can rewrite as a linear set of equations $\mathbf{A}\mathbf{v} = \mathbf{d}$ where

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \ddots & \vdots \\ 0 & -1 & 2 & -1 & 0 & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 2 \end{bmatrix}$$

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-1} \end{bmatrix}$$

and

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \end{bmatrix}$$

with $d_i = h^2 \cdot f_i$

Project 1 b)

General algorithm

We have a linear set of equations $\mathbf{A}\mathbf{v} = \mathbf{d}$

In the general case, we can express any tridiagonal matrix

$$\mathbf{A} = \begin{bmatrix} b_1 & c_1 & 0 & \cdots & \cdots & 0 \\ a_1 & b_2 & c_2 & \ddots & \ddots & \vdots \\ 0 & a_2 & b_3 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & c_{n-2} & 0 \\ 0 & \cdots & 0 & a_{n-2} & b_{n-1} & c_{n-1} \\ 0 & \cdots & \cdots & 0 & a_{n-1} & b_n \end{bmatrix}$$

just by the three vectors a , b and c , where b has length n , and a and c have length $n - 1$.

Forward substitution

Firstly, we want to eliminate the a_i 's.

$\mathbf{A}\mathbf{v} = \mathbf{d}$ gives us these equations for the case of $i = 1$ and $i = n$

$$b_1 v_1 + c_1 v_2 = d_1, \quad i = 1 \quad (1)$$

$$a_{n-1}v_{n-1} + b_nv_n = d_n, \quad i = n. \quad (2)$$

For the rest, we get

$$a_1v_1 + b_2v_2 + c_2v_3 = d_2, \quad i = 2. \quad (3)$$

$$a_{i-1}v_{i-1} + b_iv_i + c_iv_{i+1} = d_i, \quad i = 2, \dots, n-1.$$

We can then modify (3) by subtracting (1), like this

$$b_1 \cdot (3) - a_1 \cdot (1)$$

Which gives

$$\begin{aligned} (a_1v_1 + b_2v_2 + c_2v_3)b_1 - (b_1v_1 + c_1v_2)a_1 &= d_2b_1 - d_1a_1 \\ (b_2b_1 - c_1a_1)v_2 + c_2b_1v_3 &= d_2b_1 - d_1a_1. \end{aligned}$$

Notice that v_1 has been eliminated (the first lower diagonal element has been eliminated).

This can be continued further - to eliminate all the a_i 's - and is what we call *forward substitution*.

Its apparent that the vector elements get more and more complicated. To solve this, we make modified vectors and find their elements recursively. Furthermore, we ensure that the \tilde{b}_i 's are 1 by normalizing with the modified diagonal elements.

$$\begin{aligned} \tilde{b}_i &= 1 \\ \tilde{c}_1 &= \frac{c_1}{b_1} \\ \tilde{c}_i &= \frac{c_i}{b_i - \tilde{c}_{i-1}a_{i-1}} \\ \tilde{d}_1 &= \frac{d_1}{b_1} \\ \tilde{d}_i &= \frac{d_i - \tilde{d}_{i-1}a_{i-1}}{b_i - \tilde{c}_{i-1}a_{i-1}} \end{aligned}$$

Backward substitution

If we look at the coefficients defined above, we see that they give these equations for every i :

$$\begin{aligned} v_n &= \tilde{d}_n \\ v_i &= \tilde{d}_i - \tilde{c}_iv_{i+1} \end{aligned}$$

This is the *backward substitution* necessary to find the solution.

Project 1 c)**Modified algorithm**

In this case we use our general algorithm derived in Project 1 b) and simply replace our variables a_i , b_i and c_i with respectively -1 , 2 and -1 .

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & \cdots & \cdots & \cdots \\ -1 & 2 & -1 & 0 & & \\ 0 & -1 & 2 & -1 & 0 & \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & & & -1 & 2 & -1 \\ 0 & & & & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \cdots \\ \cdots \\ \cdots \\ v_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \cdots \\ \cdots \\ \cdots \\ d_n \end{bmatrix}$$

Forward substitution special case

$$\tilde{b}_i = 1\tilde{c}_1 = -\frac{1}{2}\tilde{c}_i = -\frac{1}{2 - (-1)a_{i-1}} = -\frac{1}{2 + \tilde{c}_{i-1}}\tilde{d}_1 = \frac{d_1}{2}\tilde{d}_i = \frac{d_i + \tilde{d}_{i-1}}{2 + \tilde{c}_{i-1}}$$

Backward substitution special case

In the backward substitution there will not have any differences from the one in Project 1 b), so

$$v_n = \tilde{d}_i v_i = \tilde{d}_i - \tilde{c}_i v_{i+1}$$

For the general and special algorithm the flops will run as $O(n)$. By simplifying our algorithm the number of floating points, FLOPS, decreases from **9n** to **6n**.

The CPU time for the general algorithm:

The CPU time for special algorithm:

Appendix

[Source Code](#)

Bibliography