Project 1 in FYS4150

Eirik Ramsli Hauge

September 26, 2016

Abstract

The aim of this numerical experiment is to solve the one-dimensional Poisson equation with Dirichlet boundary conditions. At first we will use the formula $\mathbf{Av} = \mathbf{p}$ and the fact that A is a tridiagnoal matrix to devlope an algorithm that we then can compare to more general methods. We found that: Write what you found!

Introduction

Theory

In this project we want to look closer at the one-dimensional Poisson's equation given as:

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial \Phi}{\partial r}\right) = -4\pi\rho(r) \tag{1}$$

where Φ is the electrostatic potential generated bya localized charge distribution $\rho(r)$. If we now do the substitution $\Phi(r) = \frac{\phi(r)}{r}$, we can rewrite equation (1) as follows:

$$\frac{\partial \phi}{\partial r^2} = -4\pi r\rho(r)$$

By letting $\phi \to u$ and $r \to x$ we end up with:

$$-u^{''}(x) = f(x), \quad x \in (0,1), \quad u(0) = u(1) = 0 \tag{2}$$

If we define the discretized approximation to u as v_i, the second derivative of u can be approximated with:

$$-\frac{v_{i+1} + v_{i-1} - 2v_i}{h^2} = f_i \qquad \text{for i = 1,...,n}$$

Where $x_i = ih$ are grid points in the interval $x_0 = 0$ to $x_{n+1} = 1$ with step length $h = 1/(n+1)$ and $f_i = f(x_i)$. With the boundary conditions $v_0 = v_{n+1} = 0$ we can see that for i = 0 we get:

$$-v_1 + 2v_0 = f_0h^2$$

For i = 1:

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

For i = 2

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

We can easily see that this gives us:

$$\underbrace{\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & -1 & 2 & -1 & 0 \\ 0 & \cdots & 0 & 0 & -1 & 2 & -1 \\ 0 & \cdots & 0 & 0 & 0 & -1 & 2 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ \vdots \\ \vdots \\ v_n \end{pmatrix}}_{\mathbf{v}} = \underbrace{\begin{pmatrix} f_1 h^2 \\ f_2 h^2 \\ f_3 h^2 \\ \vdots \\ \vdots \\ \vdots \\ f_n h^2 \end{pmatrix}}_{p}$$

By setting $f_i h^2 = p_i$ we can write this as:

$$\mathbf{A} \cdot \mathbf{v} = \mathbf{p}$$

The task Referanse asks us to make a general algorithm to solve this scenario for any values in the tridiagonal matrix. Assuming a general tridiagonal 4x4-matrix $\widetilde{\mathbf{A}}$ for simplicity we can illustrate the method for finding v as follows:

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix}$$

This gives us the following equations:

$$\begin{aligned} \mathrm{I}: v_1 b_1 + c_1 v_2 &= p_1 \\ \mathrm{II}: a_2 v_1 + b_2 v_2 + c_2 v_3 &= p_2 \\ \mathrm{III}: a_3 v_2 + b_3 v_3 + c_3 v_4 &= p_3 \\ \mathrm{IV}: a_4 v_3 + b_4 v_4 &= p4 \end{aligned}$$

We want only zeroes on the left side of the diagonal:

$$\begin{aligned} p_2^* = p_2 - p_1 \cdot \frac{a_2}{b_1} \\ &= a_2 v_1 + b_2 v_2 + c_2 v_3 - v_1 a_2 - v_2 \frac{c_1 a_2}{b_1} \\ &= v_2 (b_2 - \frac{c_1 a_2}{b_1}) + c_2 v_3 \\ &= v_2 b_2^* + c_2 v_3 \end{aligned}$$

Now we have the following matrix:

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 \\ 0 & b_2^* & c_2 & 0 \\ 0 & a_2 & b_3 & c_3 \\ 0 & 0 & a_3 & b_4 \end{pmatrix}$$

As we can see, the a_2-term disappers from the second row. Following this trail of thought we do the same for the next row.

$$\begin{aligned} p_3^* &= p_3 - p_2 \cdot \frac{a_3}{b_2^*} \\ &= v_3(b_3 - \frac{c_2 a_3}{b_2^*}) + c_3 v_4 \\ &= v_3 b_3^* + c_3 v_3 \end{aligned}$$

This gives us the general idea and we can write a general expression for both p_i^* and b_i^*:

$$p_i^* = p_i - p_{i-1}\frac{a_i}{b_{i-1}^*}, \quad \text{for i = 2,...,n,} \quad \text{and } p_1^* = p_1 \tag{3}$$

$$b_i^* = b_i - \frac{c_{i-1} a_i}{b_{i-1}^*}, \quad \text{for i = 2,...,n,} \quad \text{and } b_1^* = b_1 \tag{4}$$

Using equations (4) and (3) through the whole matrix is called forward substitution and we end up with:

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 \\ 0 & b_2^* & c_2 & 0 \\ 0 & 0 & b_3^* & c_3 \\ 0 & 0 & 0 & b_4^* \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2^* \\ p_3^* \\ p_4^* \end{pmatrix}$$

We want to find an expression for v_i and from the last row we can find a simple equation for v_4

$$b_4^* v_4 = p_4^* \Rightarrow v_4 = \frac{p_4^*}{b_4^*}$$

From the second last row we find an expression for v_3

$$b_3^* v_3 + c_3 v_4 = p_3^* \Rightarrow v_3 = \frac{p_3^* - c_3 v_4}{b_3^*}$$

Again, doing this for the next row (going downward and up) we find that v_i can be expressend in a general way as:

$$v_i = \frac{p_i^* - c_i v_{i+1}}{b_i^*}, \quad \text{for i = n-1,...,1,} \quad \text{and } v_n = \frac{p_n^*}{b_n^*} \tag{5}$$

This is called backwards substitution.
However, since matrix **A** has the same values for a, b and c for all i, we can specialize equations (3), (4) and (5). By inserting $a_i = c_i = -1$ and $b_i = 2$ in (4) we can easily see that:

$$b_i^* = \frac{i+1}{i}, \quad \text{for i = 2,...,n}, \quad \text{and } b_1^* = b_1 = 2 \tag{6}$$

$$p_i^* = p_i + \frac{p_{i-1}^*}{b_{i-1}^*}, \quad \text{for i = 2,...,n}, \quad \text{and } p_1^* = p_1 \tag{7}$$

$$v_i = \frac{p_i^* + v_{i+1}}{b_i^*}, \quad \text{for i = n-1,...,1}, \quad \text{and } v_n = \frac{p_n^*}{b_n^*} \tag{8}$$

Since our v_i is an approximation to the known solution u_i we can find the relative error by:

$$\epsilon_i = \log_{10}\left(\left|\frac{v_i - u_i}{u_i}\right|\right) \tag{9}$$

LU-decompostion

Skriv inn noe her

Programs

In this section I will present the different key parts of my program. For the full program, please visit my github.

General tridagional solver

Implementing the equations for forward substitution, (3) and (4), were done as follows:

Algorithm 1 Forward substitution

$b_1^* = b_1$
$p_1^* = p_1$
for $i = 2, n$ **do**
  $b_i^* = b_i - a_{i-1} \cdot c_{i-1}/b_{i-1}^*$
  $p_i^* = p_i - p_{i-1}^* \cdot a_i/b_{i-1}^*$
end for

The backward substitution given in equation (5) gives us:

Algorithm 2 Backward substitution

$v_n = p_n^*/b_n^*$
for $i = n-1, 1$ **do**
  $v_i = (p_i^* - c_i \cdot v_{i+1})/b_i^*$
end for

Counting number of floating point operations gives us 6 for forward substitution and 3 for backward. This gives us $9n$ FLOPS in total.

Specialized tridagional solver

Since all a and c-values are minus one and all b-values are 2 throughout the whole matrix, we don't need the general algorithm, but can use a more specialized algorithm. Our specialized algorithm is based on equation (7) and (8) (equation (6) was calculated beforehand) and was implemented as follows:

Algorithm 3 Specialized algorithm

$p_1^* = p_1$
for $i = 2, n$ **do** ▷ Forward substitution
    $p_i^* = p_i + p_{i-1}^*/d_{i-1}$
end for
$v_n = p_n^*/d_n$
for $i = n-1, 1$ **do** ▷ Backwards substitution
    $v_i = (p_i^* + v_{i+1})/d_i$
end for

Discussion

Conclusion