Project 1

FYS4150 - Computational Physics

Even S. Håland

1 Introduction

The purpose of this project is to develop an algorithm that will be used to find a numerical solution to the one-dimensional Poisson equation

$$-u''(x) = f(x), \tag{1}$$

where $x \in (0,1)$, with the Dirichlet boundary conditions $u(0) = u(1) = 0$. It will be assumed that the source term ($f(x)$) takes the form

$$f(x) = 100e^{-10x}. \tag{2}$$

The solution we obtain numerically will be compared to a closed-form solution given by

$$u(x) = 1 - \left(1 - e^{-10}\right)x - e^{-10x}. \tag{3}$$

We can easily show that this is a solution to eq. (1) by taking the first and second derivatives:

$$\begin{aligned} & u'(x) = -\left(1 - e^{-10}\right) + 10e^{-10x} \\ \Rightarrow \quad & u''(x) = -100e^{-10x} \\ \Rightarrow \quad & -u''(x) = 100e^{-10x} = f(x). \end{aligned}$$

The algorithm will be developed in two different stages; first a general one, and then a simplified one that deals with the particular problem in this project. The two versions of the algorithm will be compared in terms of number of floating point operations and CPU time.

Another important part of the project is to study the error of the numerical solution, and how it evolves as we decrease the step size in the algorithm. Finally we will also solve the equation by using library functions, and see why this might not be a good idea.

2 Discretization of the Poisson equation

To solve something numerically we need to make a discrete approximation to the problem. In this case we approximate $u(x)$ by $v(x_i) = v_i$, with $x_i = ih$ in the interval $x_0 = 0$ to $x_{n+1} = 1$. The step size is defined by $h = 1/(n+1)$. The boundary conditions are now given by $v_0 = v_{n+1} = 0$.

The second derivative is approximated by

$$u''(x) \approx \frac{v_{i+1} + v_{i-1} - 2v_i}{h^2}, \tag{4}$$

meaning that our problem can be written as

$$-\frac{v_{i+1} + v_{i-1} - 2v_i}{h^2} = f_i, \tag{5}$$

where $f_i = f(x_i)$ and $i = 1, \ldots, n$. To simplify the expression a little bit we multiply both sides by h^2, and define $\tilde{b}_i = h^2 f_i$.

Let us now write eq. (5) explicitly for some values of i (and keep in mind that $v_0 = v_{n+1} = 0$):

$$\begin{aligned} i = 1 \quad &\Rightarrow \quad -v_2 + 2v_1 = \tilde{b}_1 \\ i = 2 \quad &\Rightarrow \quad -v_3 - v_1 + 2v_2 = \tilde{b}_2 \\ i = 2 \quad &\Rightarrow \quad -v_4 - v_2 + 2v_3 = \tilde{b}_3 \\ &\vdots \\ i = n \quad &\Rightarrow \quad -v_{n-1} + 2v_n = \tilde{b}_n \end{aligned}$$

It is now relatively easy to see that this can be written as a matrix equation if we define

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \quad \text{and} \quad \tilde{\mathbf{b}} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{pmatrix}.$$

The equation can then be written as $\mathbf{A}\mathbf{v} = \tilde{\mathbf{b}}$, where $\mathbf{A}$ is the tridiagonal $n \times n$-matrix given by

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & \cdots \\ 0 & -1 & 2 & -1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{pmatrix}.$$

3 Developing the algorithm