FYS4150 Project 1:
1-dimensional Poisson equation

Marie Foss[1], **Maria Hammerstrøm**[1]

[1] Institute of Theoretical Astrophysics, University of Oslo

Abstract: We discovered ...

1. Introduction

In this project we will solve the one-dimensional Poissson equation with Dirichlet boundary conditions by rewriting it as a set of linear equations and solving these numerically in a program written in C++.The equation to be solved is:

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

and we define the discretized approximation to u as v_i with grid points $x_i = ih$ in the interval from $x_0 = 0$ to $x_{n+1} = 1$. The step length or spacing is defined as $h = 1/(n+1)$. We then have the boundary conditions $v_0 = v_{n+1} = 0$. We can approximate the second derivative of u with:

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

where $f_i = f(x_i)$.

Eq. 2 can be written as a linear set of equations of the form:

$$\mathbf{A}\mathbf{v} = \tilde{\mathbf{b}} \tag{3}$$

where $\tilde{b}_i = h^2 f_i$ and $\mathbf{A}$ is an $n \times n$ matrix of the form:

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & & -1 & 2 & -1 \\ 0 & \dots & & 0 & -1 & 2 \end{pmatrix} \tag{4}$$

(Show this!!)

We will assume that the source term is $f(x) = 100e^{-10x}$. Using the interval and boundary conditions as stated above, the above differential equation has a closed-form solution given by $u(x) = 1 - (1 - e^{-10})x - e^{-10x}$. We will compare our numerical solution with this result.

2. Solving the problem

2.1. Simple algorithm

In our case we are dealing with a simple tridiagonal matrix. We can therefore rewrite our matrix $\mathbf{A}$ in terms of one-dimensional vectors *a, b, c* of length 1:*n*. The linear equation then reads:

$$\mathbf{A} = \begin{pmatrix} b_1 & c_1 & 0 & \dots & \dots & \dots \\ a_2 & b_2 & c_2 & \dots & \dots & \dots \\ & a_3 & b_3 & c_3 & \dots & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ & & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ \dots \\ \dots \\ v_n \end{pmatrix} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \dots \\ \dots \\ \dots \\ \tilde{b}_n \end{pmatrix}. \tag{5}$$

which can be written as:

$$a_i v_{i-1} + b_i v_i + c_i v_{i+1} = \tilde{b}_i \tag{6}$$

for $i = 1, 2, \dots, n$. We want to find a simple algorithm to solve this set of equations. To do this, we can rewrite Eq. 6 by a method of elimination, which will lead to a system where there is only one unknown.

The first thing to do is to realize that Eq. 6 gives a system of three equations:

$$b_1 v_1 + c_1 v_2 = \tilde{b}_1, \qquad i = 1 \tag{7}$$

$$a_i v_{i-1} + b_i v_i + c_i v_{i+1} = \tilde{b}_i, \qquad i = 2, \dots, n-1 \tag{8}$$

$$a_n v_{n-1} + b_n v_n = \tilde{b}_n, \qquad i = n \tag{9}$$

where the boundary conditions have been applied to simplify the equations. The idea now is to subtract one row with a scalar multiple of another to eliminate variables. First we want to eliminate v_1:

(something)

Then we can follow the same approac to eliminate v_2:

(something)

The algorithm for solving this equation can be stated as follows:

(algorithm)

2.2. Using a library package

In addition to solving the linear second-order differential equation Eq. 1 using the simple algorithm described above, we want to solve the equation using Gaussian elimination and LU decomposition, then compare the results.

More coming here.

3. Analysis

3.1. ...

...

3.2. Number of floating point operations

Some of the methods described above may be more computationally heavy for a computer to do than others. One way to evaluate this is to calculate the precise number of floating point operations ("FLOPS") needed to solve the equations.

For Eq. 6 we have that: FLOPS = $2n^3$. For Gaussian elimination we have XXX and for LU decomposition XXX.

4. Conclusions

...