FYS4150 - Computational Physics
Project 1

Alex Ho

13. september 2016

1 Introduction

We will in this project solve the one-dimensional Poisson equation, given as

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

numerically, with Dirichlet boundary conditions by rewriting the Poisson equation to a set of linear equations. In this project, we will also focus on memory allocation and floating point operations in our numerical algorithm. The method section is divided in multiple parts, where each part has a different task to be solved.

2 Method

a)

We can discretize the Poisson equation as

$$\begin{aligned} -v''(x) &= f(x) \\ \implies -\frac{v_{i+1} + v_{i-1} - 2v_i}{h^2} &= f_i \end{aligned} \tag{2}$$

Where $f_i = f(x_i)$, $v_{i+1} = v(x_{i+1})$, etc. for $i = 1, 2, ..., n$. Multiply h^2 on both sides, we get the following

$$2v_i - v_{i+1} - v_{i-1} = \tilde{f}_i, \quad \tilde{f}_i = h^2 f_i \tag{3}$$

Since this is for $i = 1, 2, .., n$, one can write this as a set of linear equations

$$\begin{aligned} 2v_1 - v_2 &= \tilde{f}_1 \\ 2v_2 - v_3 - v_1 &= \tilde{f}_2 \\ 2v_3 - v_4 - v_2 &= \tilde{f}_3 \\ &\vdots \\ 2v_n - v_{n-1} &= \tilde{f}_n \end{aligned} \tag{4}$$

Here I have set $v_0 = v_{n+1} = 0$ as they are "outside"our range of values. We can clearly see a pattern here. This set of linear equation can be written in the form

$$\mathbf{A}\mathbf{v} = \tilde{\mathbf{f}} \tag{5}$$

Where

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \qquad \tilde{\mathbf{f}} = \begin{pmatrix} \tilde{f}_1 \\ \tilde{f}_2 \\ \vdots \\ \tilde{f}_n \end{pmatrix} \tag{6}$$

And the tridiagonal $n \times n$ matrix as

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & \cdots \\ 0 & -1 & 2 & -1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & -1 & 2 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{pmatrix} \tag{7}$$

We are also given the source term as $f(x) = 100e^{-10x}$ and that the closed-form solution, given the same interval and boundary conditions, is $u(x) = 1 - (1 - e^{-10})\, x - e^{-10x}$. This can easily be shown:

$$\begin{aligned} -u''(x) &= -\frac{d^2}{dx^2}\left(1 - \left(1 - e^{-10}\right) x - e^{-10x}\right) \\ &= -\frac{d}{dx}\left((1 - e^{-10}) + 10e^{-10x}\right) \\ &= -\left(-100e^{-10x}\right) = f(x) \end{aligned} \tag{8}$$

b)

Our matrix $\mathbf{A}$ can be written in terms of one-dimensional vectors a, b and c with the length $i = 1 : n$. The linear equation is then

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & \cdots & \cdots \\ a_1 & b_2 & c_2 & 0 & \cdots & \cdots \\ 0 & a_2 & b_3 & c_3 & 0 & \cdots \\ & 0 & \cdots & \cdots & \cdots & \cdots \\ & & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & & a_{n-1} & b_n \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-1} \\ v_n \end{pmatrix} = \begin{pmatrix} \tilde{f}_1 \\ \tilde{f}_2 \\ \tilde{f}_3 \\ \vdots \\ \tilde{f}_{n-1} \\ \tilde{f}_n \end{pmatrix} \tag{9}$$

We want to write an algorithm to solve the values of v_i, assuming different values for the matrix elements a_i, b_i, c_i. We also want to find the number of floating point operations (FLOPS) that is needed to solve the given algorithm.

Let us start by writing out the linear equation to be in the form similar to (5)

$$b_1 v_1 + c_1 v_2 = \tilde{f}_1 \tag{10}$$
$$a_1 v_1 + b_2 v_2 + c_2 v_3 = \tilde{f}_2 \tag{11}$$
$$a_2 v_2 + b_3 v_3 + c_3 v_4 = \tilde{f}_3 \tag{12}$$
$$\vdots$$
$$a_{n-2} v_{n-2} + b_{n-1} v_{n-1} + c_{n-1} v_n = \tilde{f}_{n-1} \tag{13}$$
$$a_{n-1} v_{n-1} + b_n v_n = \tilde{f}_n \tag{14}$$

Starting with (10) and (11), we first multiply (10) with a_1/b_1 and then subtract (11) with the new (10) (after the multiplication), we get

$$a_1 v_1 + \frac{c_1 a_1}{b_1} v_2 = \tilde{f}_1 \frac{a_1}{b_1} \tag{15}$$
$$a_1 v_1 + b_2 v_2 + c_2 v_3 - (a_1 v_1 + \frac{c_1 a_1}{b_1} v_2) = \tilde{f}_2 - \tilde{f}_1 \frac{a_1}{b_1} \tag{16}$$

Multiply b_1/a_1 on (15), and with a little rewriting we arrive to

$$b_1 v_1 + c_1 v_2 = \tilde{f}_1 \tag{17}$$
$$\tilde{b_2} v_2 + c_2 v_3 = F_2 \tag{18}$$

Where I have now defined $\tilde{b}_2 = b_2 - \frac{c_1 a_1}{b_1}$ and $F_2 = \tilde{f}_2 - \tilde{f}_1 \frac{a_1}{b_1}$. Let us now continue by looking at (12) and (19), that is

$$\tilde{b}_2 v_2 + c_2 v_3 = F_2 \tag{19}$$

$$a_2 v_2 + b_3 v_3 + c_3 v_4 = \tilde{f}_3 \tag{20}$$

Doing the same trick as we did previously, we multiply $a_2/\tilde{b}_2$ on equation (19) and then subtract that on (20). Rewriting a little bit we end up with

$$\tilde{b}_2 v_2 + c_2 v_3 = F_2 \tag{21}$$

$$\tilde{b}_2 v_3 + c_3 v_4 = F_3 \tag{22}$$

Here I have defined $\tilde{b_3} = b_3 - \frac{c_2 a_2}{\tilde{b_2}}$ and $F_3 = \tilde{f}_3 - F_2 \frac{a_2}{\tilde{b_2}}$. One can continue to do this, but we can clearly see a pattern here, which results to these general expressions

$$\tilde{b}_i = b_i - \frac{c_{i-1} a_{i-1}}{\tilde{b}_{i-1}} \tag{23}$$

$$F_i = \tilde{f}_i - F_{i-1} \frac{a_{i-1}}{\tilde{b}_{i-1}}, \qquad \text{for } i = 2, ...n \tag{24}$$

This is known as *Forward Substitution*. For $i = 1$, we need to explicitly define that $\tilde{bn}_1 = b_1$ and $F_1 = \tilde{f}_1$. This is mostly because values like c_0 and a_0 are outside our interval.[1]. The number of floating point operations (FLOPS) for these two steps combined is 6 FLOPS. Both steps do one subtraction, one multiplication and one division, which means that each of them do 3 FLOPS, and the total will therefore be 6 FLOPS.

The next step is to find an algorithm to find the values of v_i. From the calculations above, our linear equations are as following

$$b_1 v_1 + c_1 v_2 = \tilde{f}_1 \tag{25}$$

$$\tilde{b}_2 v_2 + c_2 v_3 = F_2 \tag{26}$$

$$\tilde{b}_3 v_3 + c_3 v_4 = F_3 \tag{27}$$

$$\vdots$$

$$\tilde{b}_{n-1} v_{n-1} + c_{n-1} v_n = F_{n-1} \tag{28}$$

$$\tilde{b}_n v_n = F_n \tag{29}$$

[1]Not to be confused with the index notation in C++ and Python, where intervals goes as $x_0, x_1, ..., x_{n-1}$

From (29) we have that $v_n = F_n/\tilde{b}_2$, which is the "initial"value of v and we will have to calculate backwards to get all the values of v_i. Equation (28) gives us the relation:

$$v_{n-1} = \frac{1}{\tilde{b}_{n-1}} \left(F_{n-1} - c_{n-1} v_n\right)$$

In terms of i's, this will become

$$v_{i-1} = \frac{1}{\tilde{b}_{i-1}} \left(F_{i-1} - c_{i-1} v_i\right), \quad \text{for } i = n, n-1, ..., 2 \tag{30}$$

This is known as *Backward Substitution*. The number of FLOPS in this case will be 3 FLOPS.Like Forward substitution, we have one subtraction, one multiplication and one division. The number of FLOPS for both Forward substitution and Backward substitution is 9 FLOPS. We also have to keep in mind that we do these calculations n times, so the total number of FLOPS for all n equations is $9n$ FLOPS.

c)

We now use the fact that our matrix $\mathbf{A}$ has identical matrix elements along the diagonal and the non-diagonal elements are also identical (but different from the diagonal one). With our specific tri-diagonal matrix $\mathbf{A}$, we will specialize our algorithm from the previous task to solve our linear equation.

Let us first look at the forward substitution algorithm for $\tilde{b}_i$, we had

$$\tilde{b}_i = b_i - \frac{c_{i-1} a_{i-1}}{\tilde{b}_{i-1}} \tag{31}$$

With the non diagonal elements as $a_i = c_i = -1$ and $b_i = 2$. With the initial value $\tilde{b}_1 = b_1 = 2$, we can write out every $\tilde{b}_i$ value, but after a few steps, one

can see a pattern and we end up with

$$\begin{aligned}
\tilde{b}_2 &= 2 - \frac{1}{2} = \frac{3}{2} \\
\tilde{b}_3 &= 2 - \frac{1}{\frac{3}{2}} = \frac{4}{3} \\
\tilde{b}_4 &= 2 - \frac{1}{\frac{4}{3}} = \frac{6}{5} \\
&\vdots \\
\tilde{b}_i &= \frac{i+1}{i}, \quad \text{for } i = 1, 2, ..., n
\end{aligned} \tag{32}$$

The next step in the forward substitution algorithm is then

$$\begin{aligned}
F_i &= \tilde{f}_i - F_{i-1}\frac{a_{i-1}}{\tilde{b}_{i-1}} \\
\implies F_i &= \tilde{f}_i + F_{i-1}\left(\frac{i}{i+1}\right), \quad \text{for } i = 2, ..., n
\end{aligned} \tag{33}$$

Where the initial value is $F_1 = \tilde{f}_1$. Finally, the backward substitution algorithm can be simplified to

$$\begin{aligned}
v_{i-1} &= \frac{1}{\tilde{b}_{i-1}}(F_{i-1} - c_{i-1}v_i) \\
\implies v_{i-1} &= \frac{i}{i+1}(F_{i-1} + v_i), \quad \text{for } i = n, n-1, ..., 2
\end{aligned} \tag{34}$$

The number of FLOPS for eq. (32) is 2, one addition and one division. Both (33) and (34) uses 4 FLOPS, with two additions, one multiplication and one division. So the total number of FLOPS should be 10. However, because $\tilde{b}_i$ is already mixed into eq. (33) and (34), we do not actually have to calculate (32) at all. So the the number of FLOPS is then reduced to 8. Again, since we are solving these equations n times, total number of FLOPS is then $8n$ FLOPS.

d)

3 Implementation

I have used both C++ and Python to solve this project. The strategy is to use C++ to develop and algorithm, do the calculations and save the results

to a text file. Once I have the results, I use Python to plot the values that was calculated in C++. All files for this project can be found in the GitHub repository `https://github.com/AHo94/FYS3150_Projects/tree/master/Project1`.

a)

4 Results

5 Conclusion

6 References

Hjort-Jensen,M. 2015. *Computational Physics*, accessible at course GitHub repository `https://github.com/CompPhysics/ComputationalPhysics/tree/master/doc/Lectures`, 551 pages.