Project 1. FYS3150

Shafa Aria

September 16, 2013

Introduction: In this project we shall familiarize ourselves witht the various matrix operations, DMA and using the various libraries. The project itself evolves around Poisson equation and the numerical implementation of it using linear algebra packages for C++. We will in the following solve the linear equations and find numerical solutions for the equation and compare it with the analytical solution. This project was a collaboration with Abas Omar.

(a) Our equation may be rewritten by multiplying with h squared:

$$2v_i - v_{i+1} - v_{i-1} = f_i h^2$$

We recognise the RHS as given to us being $b_i = h^2 f_i$ this is a simple difference equation that can be rewritten as the product of 2 matrices where $\mathbf{A}$ contains the constant values and $\mathbf{v}$ the function values.

$$2 \cdot v_i - 1 \cdot v_{i+1} - 1 \cdot v_{i-1} = f_i h^2$$

$$\mathbf{Av} = \begin{bmatrix} a_1 & a_2 & a_3 & \dots & \dots & a_n \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \dots \\ \dots \\ v_n \end{bmatrix} \tag{1}$$

$$= \begin{bmatrix} a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + \dots + a_n \cdot v_n \end{bmatrix} \tag{2}$$

(b) We realise that our matrix contains only three constants across the diagonal and thus we can write them as vectors instead of having the entire matrix written. We will then write an algorithm that will collect all the contributing factors.

$$\mathbf{a} = \begin{bmatrix} a_2 \\ a_3 \\ a_4 \\ \dots \\ \dots \\ a_n \end{bmatrix} \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ \dots \\ b_n \end{bmatrix} \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \dots \\ \dots \\ c_{n-1} \end{bmatrix} \tag{3}$$

We will use forward substitution to remove the lower diagonal and then backward substition to remove the upper diagonal. For the forward we find:

$$\tilde{b}_{i+1} = \tilde{b}_{i+1} - \tilde{b}_i \frac{a_{i+1}}{b_i} \quad \text{for } i = 1, \dots, n-1$$

$$b_{i+1} = b_{i+1} - \frac{a_{i+1} \cdot c_i}{b_i} \quad \text{for } i = 1, \dots, n-1$$

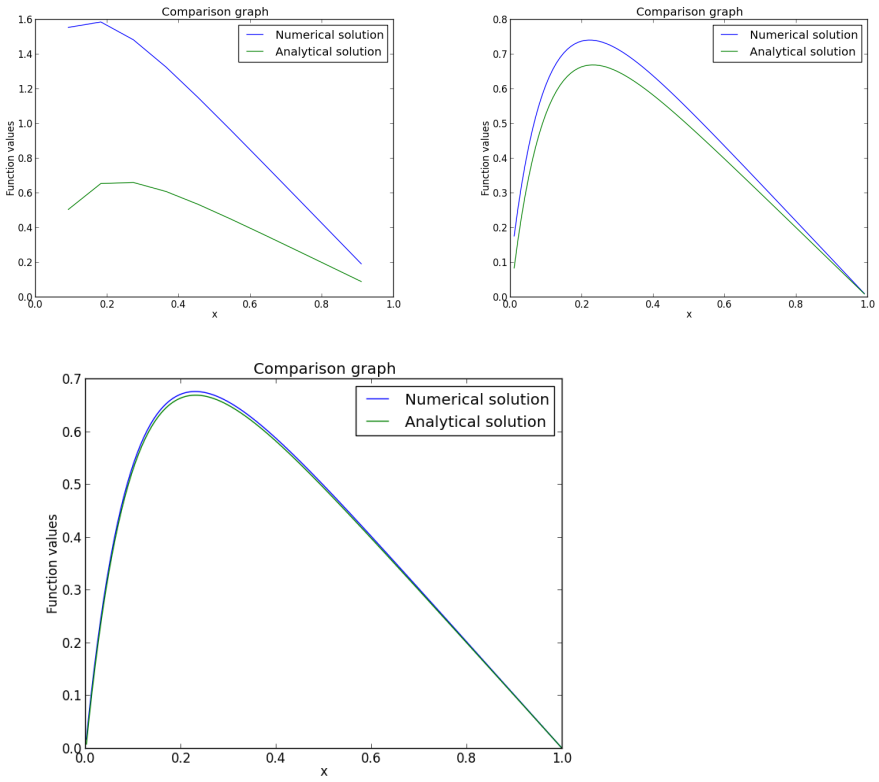
And for the backwards:

$$\tilde{b}_{i-1} = \tilde{b}_{i-1} - c_{i-1} \cdot \frac{\tilde{b}_i}{b_i} \quad \text{for } i = n, \dots, 1$$

And thus:

$$x_i = \frac{\tilde{b}_i}{b_i}$$

For total FLOP we count our iterations: $2 \cdot (3 \cdot (n-1))$ for forward,$3n+n$ for backwards. Total of $10n-6$ FLOP and arithmetic complexity of $O(n^2)$ for this algorithm. For comparison LU and Gaussian Elimination requires $2/3n^3$ has arithmetic complexity of $O(n^3)$.



As we can see shorter steps gives higher accuracy, and thus our numerical calculation approximates the analytical solution better. From top left: $n = 10$, $n = 100$ and at the bottom $n = 1000$. Albeit as we can see despite $n = 1000$ we can clear see there's a difference between the two, but as we shorten our step the curve lies more and more onto the analytical one.

(c) The expected results is of course that the accuracy will increase, but we see that the relative error is approximately porportional to the log(h). The order of the relative error is therefore approximately the same as h.

$$\begin{bmatrix} \log(\mathrm{h}) = -1 & \mathrm{epsi} = 0.728827503126 \\ \log(\mathrm{h}) = -2 & \mathrm{epsi} = 0.088645078904 \\ \log(\mathrm{h}) = -3 & \mathrm{epsi} = 0.009085353860 \\ \log(\mathrm{h}) = -4 & \mathrm{epsi} = 0.000910799789 \\ \log(\mathrm{h}) = -5 & \mathrm{epsi} = 9.11346953\mathrm{e}-05 \end{bmatrix} \tag{4}$$

(d) As for d we are supposed to compare our results with the LU decomposition for the various steps and see how long the computation takes. As it is now we could not get our code to work, there's a mismatch between the armadillo package and the vector/matrix opreations. Albeit the code is included.

(e) See the code.

Comment: the link for the github:

https://github.com/Ace-/Project-1