

# Project 1 - FYS3150

René Ask and Benedicte Nyheim  
(Dated: August 30, 2019)

## I. METHOD

We're going to solve the differential equation

$$-u''(x) = f(x), \quad x \in (0, 1), \quad u(0) = u(1) = 0. \quad (1)$$

To this end, we'll derive an approximation to the second derivative of  $u(x)$  before we estimate the total error of the approximation.

### A. Derivation of the approximation scheme and an upper-bound on its total error

We start by Taylor expanding about some point  $x$

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(\xi_1), \quad (2)$$

and

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(\xi_2), \quad (3)$$

where  $\xi_1 \in (x, x+h)$  and  $\xi_2 \in (x-h, x)$ . Adding the two equations and rearranging a bit we get

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{h^2}{4!}[f^{(4)}(\xi_1) - f^{(4)}(\xi_2)]. \quad (4)$$

To find the signed truncation error, we rewrite the equation as

$$f''(x) - \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} = -\frac{h^2}{24}[f^{(4)}(\xi_1) - f^{(4)}(\xi_2)]. \quad (5)$$

Before we proceed with the error analysis, we might as well pick  $\xi \in (x-h, x+h)$  such that  $f(\xi) = \max\{f(\xi_1), f(\xi_2)\}$  and multiply it by a factor 2 at the RHS, that is

$$f''(x) - \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} = -\frac{h^2}{12}f(\xi). \quad (6)$$

Due to the fact that computers are unable to represent numbers exactly, the computed values can be written as  $\bar{f}(x+h) = f(x+h)(1+\epsilon_1)$ ,  $\bar{f}(x) = f(x)(1+\epsilon_2)$  and  $\bar{f}(x-h) = f(x-h)(1+\epsilon_3)$ . An upper-bound estimate of the global total error, that is, the truncation error and the error due to loss of precision is

$$\begin{aligned} \epsilon &= |f'(x) - \bar{f}'(x)| \\ &= \left| f''(x) - \frac{\bar{f}(x+h) - 2\bar{f}(x) + \bar{f}(x-h)}{h^2} \right| \\ &= \left| f''(x) - \frac{f(x+h)(1+\epsilon_1) - 2f(x)(1+\epsilon_2) + f(x-h)(1+\epsilon_3)}{h^2} \right| \\ &= \left| f''(x) - \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{f(x+h)\epsilon_1 - 2f(x)\epsilon_2 + f(x-h)\epsilon_3}{h^2} \right| \\ &\leq \left| -\frac{h^2}{12}f(\xi) - \frac{\epsilon_1 - 2\epsilon_2 + \epsilon_3}{h^2} \max_{\zeta \in (x-h, x+h)} f(\zeta) \right| \\ &\leq \frac{h^2}{12} \max_{\xi \in (x-h, x+h)} |f^{(4)}(\xi)| + \frac{|\epsilon_1| + 2|\epsilon_2| + \epsilon_3}{h^2} \max_{\zeta \in (x-h, x+h)} |f(\zeta)| \\ &\leq \frac{h^2}{12} \max_{\xi \in (0,1)} |f^{(4)}(\xi)| + \frac{4\epsilon_M}{h^2} \max_{\zeta \in (0,1)} |f(\zeta)|. \end{aligned} \quad (7)$$

Furthermore, we set  $\epsilon_M = \max\{|\epsilon_1|, |\epsilon_2|, |\epsilon_3|\}$ . Now let  $M_1 \equiv \max_{\xi \in (0,1)} |f^{(4)}(\xi)|$  and  $m_2 \equiv \max_{\zeta \in (0,1)} |f(\zeta)|$ . Then the upper-bound estimate of the total error can neatly be written as

$$\epsilon \leq \frac{h^2}{12} M_1 + \frac{4\epsilon_M}{h^2} M_2. \quad (8)$$

Solving  $d\epsilon/dh = 0$  with respect to  $h$  yields the optimal choice of step-size  $h^*$  given as

$$h^* = \left( \frac{48\epsilon_M M_2}{M_1} \right)^{1/4}. \quad (9)$$

We need to determine  $M_1$  and  $M_2$  to estimate  $h^*$  for our particular closed-form solution to the differential equation. This function is

$$f(x) = 1 - (1 - e^{-10})x - e^{-10x}, \quad (10)$$

and solving  $df/dx = 0$  yields

$$\zeta = -\frac{\ln(1 - e^{-10}) - \ln(10)}{10}, \quad (11)$$

such that  $M_2 = |f(\zeta)|$ . Furthermore,  $f^{(4)}(x) = -10^4 \exp(-10x)$ , hence we see that  $M_1 \leq \lim_{\xi \rightarrow 0} |f^{(4)}(\xi)|$ . For simplicity, we might as well just set  $M_1 = |f^{(4)}(0)|$ . Then we obtain

$$h^* \approx \left( \frac{48\epsilon_M |f(\zeta)|}{|f^{(4)}(0)|} \right)^{1/4} \approx 4 \times 10^{-5}, \quad (12)$$

where we set  $\epsilon_M = 10^{-15}$ . For our purposes, it's more convenient to express it in terms of the logarithm

$$\log_{10}(h^*) \approx -4.4. \quad (13)$$

## B. Approximate solution of the differential equation by the Thomas algorithm

We'll approximate this differential equation by a function  $v(x) \approx u(x)$ . By the derivation above, its clear that we can write the differential eq. for each  $x_i$  as

$$-\frac{-v_{i+1} + v_{i-1} - 2v_i}{h^2} = f_i, \quad i = 1, 2, \dots, n, \quad (14)$$

which may be rearranged into

$$2v_i - v_{i+1} - v_{i-1} = f_i h^2 \equiv q_i. \quad (15)$$

From (15) we can write

$$\begin{pmatrix} 2v_1 - v_2 \\ -v_1 + 2v_2 - v_3 \\ \vdots \\ 2v_n - v_{n-1} \end{pmatrix} = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ \vdots & & & & \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{pmatrix} \quad (16)$$

To this end we'll first develop a general algorithm based on forward- and backward substitution to solve the equation  $A\mathbf{x} = \mathbf{q}$ . Performing one step of Gaussian elimination on the following matrix yields

$$A = \begin{pmatrix} b_1 & c_1 & 0 & \cdots & \cdots & \cdots \\ a_1 & b_2 & c_2 & 0 & \cdots & \cdots \\ 0 & a_2 & b_3 & 0 & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n-2} & b_{n-1} & c_{n-1} \\ 0 & 0 & \cdots & 0 & a_{n-1} & b_n \end{pmatrix} \sim \begin{pmatrix} b_1 & c_1 & 0 & \cdots & \cdots & \cdots \\ 0 & b_2 - \frac{a_1}{b_1}c_1 & c_2 & 0 & \cdots & \cdots \\ 0 & a_2 & b_3 & 0 & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n-2} & b_{n-1} & c_{n-1} \\ 0 & 0 & \cdots & 0 & a_{n-1} & b_n \end{pmatrix}. \quad (17)$$

Defining  $\tilde{b}_2 \equiv b_2 - (a_1/b_1)c_1$ , we can perform another step

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & \cdots & \cdots \\ 0 & \tilde{b}_2 & c_2 & 0 & \cdots & \cdots \\ 0 & a_2 & b_3 & 0 & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n-2} & b_{n-1} & c_{n-1} \\ 0 & 0 & \cdots & 0 & a_{n-1} & b_n \end{pmatrix} \sim \begin{pmatrix} b_1 & c_1 & 0 & \cdots & \cdots & \cdots \\ 0 & \tilde{b}_2 & c_2 & 0 & \cdots & \cdots \\ 0 & 0 & b_3 - \frac{a_2}{\tilde{b}_2}c_2 & 0 & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n-2} & b_{n-1} & c_{n-1} \\ 0 & 0 & \cdots & 0 & a_{n-1} & b_n \end{pmatrix}, \quad (18)$$

and at this point it's pretty obvious that the general pattern for the forward-substitution is

$$b_i = b_i - \frac{a_i}{b_i}c_i, \quad \text{for } i = 2, 3, \dots, n, \quad (19)$$

$$q_i = y_i - \frac{a_i}{b_i}y_i, \quad \text{for } i = 2, 3, \dots, n, \quad (20)$$

where the second equation is simply performing the exact same operation on the RHS of the equation. Once this process is completed, we need to perform back-substitution to find  $\mathbf{x}$ . At this point our equation will look as follows:

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & \cdots & \cdots \\ 0 & \tilde{b}_2 & c_2 & 0 & \cdots & \cdots \\ 0 & 0 & \tilde{b}_3 & 0 & \cdots & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \tilde{b}_{n-1} & c_{n-1} \\ 0 & 0 & \cdots & 0 & 0 & \tilde{b}_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1x_1 + c_1x_2 \\ \tilde{b}_2x_2 + c_2x_3 \\ \tilde{b}_3x_3 + c_3x_4 \\ \vdots \\ \tilde{b}_{n-1}x_{n-1} + c_{n-1}x_n \\ \tilde{b}_nx_n \end{pmatrix} = \begin{pmatrix} \tilde{q}_1 \\ \tilde{q}_2 \\ \tilde{q}_3 \\ \vdots \\ \tilde{q}_{n-1} \\ \tilde{q}_n \end{pmatrix}, \quad (21)$$

so the general algorithm for the back-substitution is

$$x_i = \frac{q_i}{b_i}, \quad \text{for } i = n, \quad (22)$$

$$x_i = \frac{q_i - c_i x_{i+1}}{b_i}, \quad \text{for } i = n-1, n-2, \dots, 1. \quad (23)$$

For our particular matrix, we know that  $b_i = 2$  and  $c_i = a_i = -1$ , so we can specialize the algorithm. To see this, we first motivate the formula by

$$b_1 = b = 2, \quad (24)$$

$$b_2 = b - 1/b_1 = 2 - \frac{1}{2} = \frac{3}{2}, \quad (25)$$

$$b_3 = b - 1/b_2 = 2 - \frac{1}{3/2} = 2 - \frac{2}{3} = \frac{4}{3}, \quad (26)$$

$$\vdots \quad (27)$$

$$b_i = \frac{i+1}{i}, \quad \text{for } i = 1, 2, \dots, n. \quad (28)$$

To this end we will develop an algorithm based on the LU-decomposition  $A = LU$ :

$$A = \begin{pmatrix} b_1 & c_1 & 0 & \cdots & \cdots & \cdots \\ a_1 & b_2 & c_2 & 0 & \cdots & \cdots \\ 0 & a_2 & b_3 & 0 & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n-2} & b_{n-1} & c_{n-1} \\ 0 & 0 & \cdots & 0 & a_{n-1} & b_n \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \ell_2 & 1 & \cdots & \cdots & \cdots & 0 \\ 0 & \ell_3 & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \ell_{n-1} & 1 & 0 \\ 0 & 0 & \cdots & \cdots & \ell_n & 1 \end{pmatrix} \begin{pmatrix} d_1 & u_1 & \cdots & \cdots & \cdots & 0 \\ 0 & d_2 & u_2 & \cdots & \cdots & 0 \\ 0 & 0 & d_3 & u_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & d_{n-1} & u_{n-1} \\ 0 & 0 & \cdots & \cdots & 0 & d_n \end{pmatrix} = LU, \quad (29)$$

and performing matrix multiplication we get

$$LU = \begin{pmatrix} d_1 & u_1 & \cdots & \cdots & \cdots & 0 \\ \ell_2 d_1 & \ell_2 u_1 + d_2 & u_2 & \cdots & \cdots & 0 \\ 0 & \ell_3 d_2 & \ell_3 u_2 + d_3 & u_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \ell_{n-1} d_{n-2} & \ell_{n-1} u_{n-2} + d_{n-1} & u_{n-1} \\ 0 & 0 & \cdots & \cdots & \ell_n d_{n-1} & \ell_n u_{n-1} + d_n \end{pmatrix}, \quad (30)$$

which yields the following general relations:

$$b_i = d_i, \quad c_i = u_i, \quad \text{for } i = 1, \quad (31)$$

$$\ell_i = \frac{a_{i-1}}{d_{i-1}}, \quad \text{for } 1 < i < n, \quad (32)$$

$$d_i = b_i - \ell_i u_{i-1}, \quad \text{for } 1 < i < n, \quad (33)$$

$$\ell_n = \frac{a_{n-1}}{d_{n-1}}, \quad d_n = b_n - \ell_n u_{n-1}, \quad \text{for } i = n. \quad (34)$$

requiring  $2n$  multiplications and  $n$  additions. In other words, the total floating point operations involved in finding  $LU$  is  $3n$ .

We can then write  $A\mathbf{v} = LU\mathbf{v} = L\mathbf{y} = \tilde{\mathbf{b}}$  where  $\mathbf{y} \equiv U\mathbf{v}$ . Explicitly, we can write this as

$$L\mathbf{y} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \ell_2 & 1 & \cdots & \cdots & \cdots & 0 \\ 0 & \ell_3 & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \ell_{n-1} & 1 & 0 \\ 0 & 0 & \cdots & \cdots & \ell_n & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \ell_2 y_1 + y_2 \\ \ell_3 y_2 + y_3 \\ \vdots \\ \ell_{n-1} y_{n-2} + y_{n-1} \\ \ell_n y_{n-1} + y_n \end{pmatrix} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{pmatrix} \quad (35)$$

which yields the following procedure:

$$y_1 = \tilde{b}_1, \quad (36)$$

$$y_i = \tilde{b}_i - \ell_i y_{i-1}, \quad \text{for } i = 2, 3, \dots, n. \quad (37)$$

giving  $n - 1$  multiplications and  $n - 1$  additions. Thus the added computational cost is  $2(n - 1)$ .

Finally, to determine  $\mathbf{v}$ , we perform back-substitution by solving  $U\mathbf{v} = \mathbf{y}$ . Writing it out explicitly yields

$$\begin{pmatrix} d_1 & u_1 & \cdots & \cdots & \cdots & 0 \\ 0 & d_2 & u_2 & \cdots & \cdots & 0 \\ 0 & 0 & d_3 & u_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & d_{n-1} & u_{n-1} \\ 0 & 0 & \cdots & \cdots & 0 & d_n \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} d_1 v_1 + u_1 v_2 \\ d_2 v_2 + u_2 v_3 \\ \vdots \\ \vdots \\ d_{n-1} v_{n-1} + u_{n-1} v_n \\ d_n v_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix}, \quad (38)$$

which yields the following procedure:

$$v_n = \frac{y_n}{d_n}, \quad (39)$$

$$v_i = \frac{y_i - u_i v_{i+1}}{d_i}, \quad \text{for } i = n - 1, n - 2, \dots, 1. \quad (40)$$

Here we got  $2n - 1$  multiplications and  $n - 1$  additions, so the number of floating point operations are  $3n - 2$ . Putting all of these together we get  $4(2n - 1) \approx 8n$  floating point operations.

Now, assuming that  $b_1 = b_2 = \cdots = b_n \equiv b$  and  $a_1 = c_1, a_2 = c_2, \dots, a_{n-1} = c_{n-1}$ . Furthermore, assume  $a_1 = a_2 = \cdots = a_n \equiv a$  and  $c_1 = c_2 = \cdots = c_n \equiv c$ . But since  $a = c$ , we can ignore the last assumption. These

assumptions implies certain simplifications of the algorithm for LU-decomposition:

$$d_1 = b, \quad u_1 = c = a, \quad (41)$$

$$u_i = c_i = a, \quad \text{for } 1 < i \leq n, \quad (42)$$

$$\ell_i = \frac{a_{i-1}}{d_{i-1}} = \frac{a}{d_{i-1}}, \quad \text{for } 1 < i \leq n, \quad (43)$$

$$d_i = b_i - \ell_i u_i = b - \ell_i a = b - \frac{a^2}{d_{i-1}} \quad \text{for } 1 < i \leq n. \quad (44)$$