FYS3150 Computational Physics - Project 1

Nicholas Karlsen

A look on two similar algorithms for solving an inhomogeneous second-order differential equation with a known analytical solution and comparing their efficiency to a standard LU-decomposition method. Found that the algorithms allow for greater precision and faster run-times.

INTRODUCTION

When assuming spherical symmetry, the Poisson equation, eqn. 1

$$\nabla^2 \mathbf{\Phi} = -4\pi\rho(\mathbf{r}) \tag{1}$$

can be reduced to the 1-dimensional second order inhomogeneous differential equation, eqn. 2, by assuming spherical symmetry and scaling the problem.

$$\frac{d^2u(x)}{dx^2} = f(x) \tag{2}$$

A numerical solver for this equation not only allows us to solve the Poisson equation, but several other systems as well. In this report i have looked at one such solver, in both its general and specialized form and compared its speed to a solver which employs the LU-decomposition method in scipy and found that the solver was faster, but more importantly, allowed for a solution closer to what is allowed by the machine precision.

THEORY, ALGORITHMS AND METHODS

We have second order inhomogeneous differential equation, Eqn. 3

$$\frac{d^2u(x)}{dx^2} = f(x) \tag{3}$$

With boundary conditions $\ddot{u}(x) = f(x)$, $u(0) = u(1) = 0$ and $x \in (0, 1)$.

If we let $f(x) = 100e^{-10x}$, then it can be shown analytically that the differential equation has a solution Eqn. 4

$$u(x) = 1 - (1 - e^{-10})x - e^{-10x} \tag{4}$$

This problem can also be solved numerically by discretization. By the use of taylor expansions, it can be shown that there is a discrete, iterative solution in the form Eqn. 5 [1]

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

where $h = 1/(n-1)$ is the step size for n points and V_i is the discretized form of $u(x)$.

This problem can be written in the form

$$A\vec{v} = \vec{b} \tag{6}$$

Where A $\in R^{n\times n}$ is a tridiagonal matrix with diagonal elements 2 and -1 (Eqn. 9) and $\vec{b} = h^2(f_1, \ldots, f_n)$.

$$A = \begin{bmatrix} 2 & -1 & 0 & \ldots & \ldots & 0 \\ -1 & 2 & -1 & 0 & \ldots & 0 \\ 0 & -1 & 2 & -1 & \ldots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & -1 \\ \ldots & \ldots & \ldots & \ldots & -1 & 2 \end{bmatrix} \tag{7}$$

By matrix multiplication, when we multiply the i-th row of A by $\vec{v}$, we get

$$\begin{aligned} -1v_{i-1} + 2v_i - 1v_{i+1} &= -\left(v_{i+1} + v_{i-1} - 2v_i\right) \\ &= h^2 f_i \end{aligned} \tag{8}$$

Which is equivalent to the discretized differential equation Eqn. 5, showing that the differential equation can be solved as a linear algebra problem.

A can be written more generally as

$$A = \begin{bmatrix} b_1 & c_1 & 0 & \ldots & \ldots & 0 \\ a_2 & b_2 & c_2 & 0 & \ldots & 0 \\ 0 & a_3 & b_3 & c_3 & \ldots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & c_{n-1} \\ \ldots & \ldots & \ldots & \ldots & a_n & b_n \end{bmatrix} \tag{9}$$

RESULTS AND DISCUSSIONS

The implementation of the code discussed in the previous section can be found on my github: https://github.com/nicholaskarlsen/FYS3150.

CONCLUSIONS

[1] M. Hjorth-Jensen, Computational Physics - Lecture Notes 2015, (2015).

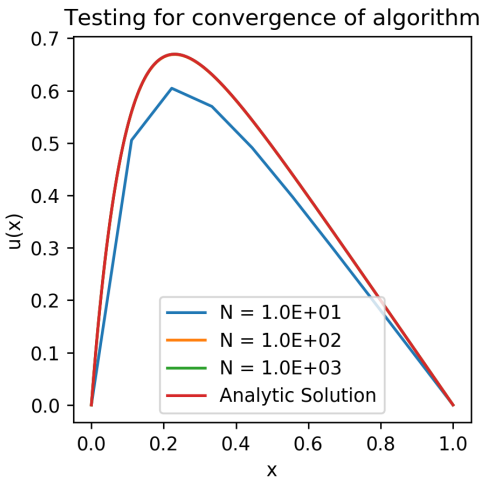

FIG. 1. A plot of the numeric solution for different N and the analytic solution.

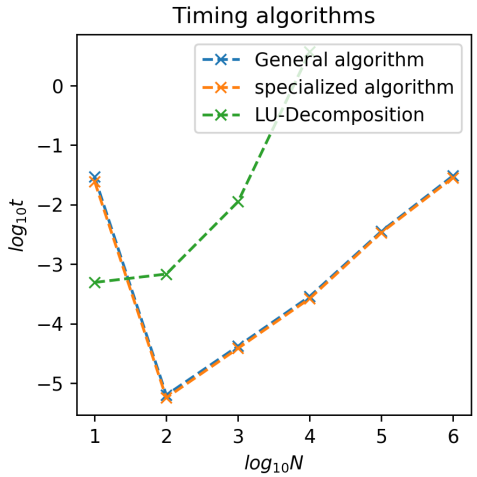

FIG. 2. The average execution time of 10 function calls on my computer

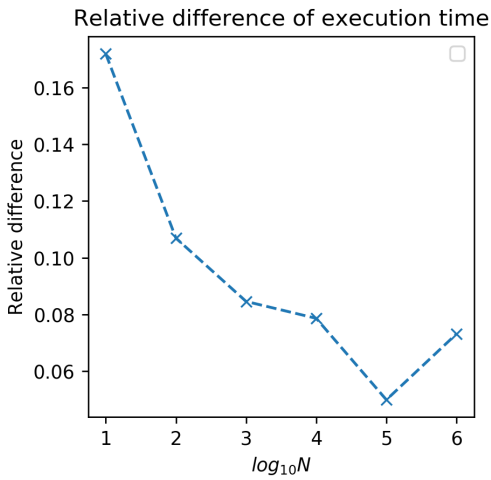

FIG. 3. Percentage difference between the execution times of the general and specialized algorithms on my computer

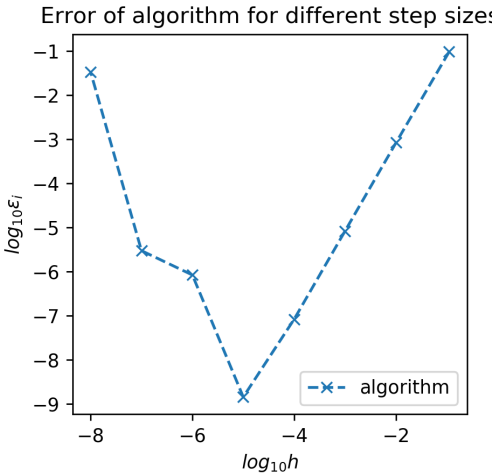

FIG. 4. How the error evolves for smaller step sizes when comparing the general algorithm to the analytic solution