

# Project 1

S. O. Mebody,<sup>1</sup> A. N. Ybody,<sup>2</sup> and N. O. Body<sup>3</sup>

<sup>1</sup>*Department of Procrastination, University of Akiva*

<sup>2</sup>*Department of Odd Travels, University of Anoot*

<sup>3</sup>*Department of Even Travels, University of Ambria*

We present a method for solving one-dimensional Poisson equation with Dirichlet boundary conditions. We find before how to rewrite the differential equation as a linear set of equations and then we solve it using gaussian elimination and LU decomposition. There is a comparison between the two methods thanks to the analysis of the elapsed time and the number of flops. We study the loss of numerical precision decreasing the step length.

## INTRODUCTION

The project shows how it is possible to solve numerically the one-dimensional Poisson using linear algebra. Furthermore the

## METHODS AND ALGORITHMIS

Many important differential equations in Science can be written as linear second-order differential equations

$$\frac{d^2y}{dx^2} + k^2(x)y = f(x),$$

where  $f$  is normally called the inhomogeneous term and  $k^2$  is a real function.

A classical equation from electromagnetism is Poisson's equation. The electrostatic potential  $\Phi$  is generated by a localized charge distribution  $\rho(\mathbf{r})$ . In three dimensions it reads

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

With a spherically symmetric  $\Phi$  and  $\rho(\mathbf{r})$  the equations simplifies to a one-dimensional equation in  $r$ , namely

$$\frac{1}{r^2} \frac{d}{dr} \left( r^2 \frac{d\Phi}{dr} \right) = -4\pi\rho(r),$$

which can be rewritten via a substitution  $\Phi(r) = \phi(r)/r$  as

$$\frac{d^2\phi}{dr^2} = -4\pi r\rho(r).$$

The inhomogeneous term  $f$  or source term is given by the charge distribution  $\rho$  multiplied by  $r$  and the constant  $-4\pi$ .

We will rewrite this equation by letting  $\phi \rightarrow u$  and  $r \rightarrow x$ . The general one-dimensional Poisson equation reads then

$$-u''(x) = f(x).$$

In this project we will solve the one-dimensional Poisson equation with Dirichlet boundary conditions by rewriting it as a set of linear equations.

To be more explicit we will solve the equation

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

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 have then the boundary conditions  $v_0 = v_{n+1} = 0$ . We approximate the second derivative of  $u$  with

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

where  $f_i = f(x_i)$ . We can rewrite this equation as a linear set of equations of the form

$$\mathbf{A}\mathbf{v} = \tilde{\mathbf{b}},$$

where  $\mathbf{A}$  is an  $n \times n$  tridiagonal matrix which we rewrite as

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & & -1 & 2 & -1 \\ 0 & \dots & & 0 & -1 & 2 \end{bmatrix},$$

and  $\tilde{b}_i = h^2 f_i$ .

In our case we will assume that the source term is  $f(x) = 100e^{-10x}$ , and keep the same interval and boundary conditions. Then the above differential equation has a closed-form solution given by  $u(x) = 1 - (1 - e^{-10})x - e^{-10x}$  (convince yourself that this is correct by inserting

FIG. 1. Our results could not be better

the solution in the Poisson equation). We will compare our numerical solution with this result in the next exercise.

We can rewrite our matrix  $\mathbf{A}$  in terms of one-dimensional vectors  $a, b, c$  of length  $1 : n$ . Our linear equation reads

$$\mathbf{A} = \begin{bmatrix} b_1 & c_1 & 0 & \dots & \dots & \dots \\ a_1 & b_2 & c_2 & \dots & \dots & \dots \\ & a_2 & b_3 & c_3 & \dots & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ & & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & & a_{n-1} & b_n \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ \dots \\ \dots \\ v_n \end{bmatrix} = \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \dots \\ \dots \\ \dots \\ \tilde{b}_n \end{bmatrix}.$$

We can compute the relative error in the data set  $i = 1, \dots, n$ , by setting up

$$\epsilon_i = \log_{10} \left( \left| \frac{v_i - u_i}{u_i} \right| \right),$$

as function of  $\log_{10}(h)$  for the function values  $u_i$  and  $v_i$ . For each step length extract the max value of the relative error. Try to increase  $n$  to  $n = 10^7$ . Make a table of the results and comment your results. You can use either the algorithm from b) or c).

To compute the elapsed time in c++ you can use the following statements

```
...
#include "time.h" //
you have to include the time.h header
```

```
int main()
{
    // declarations of variables
    ...
    clock_t start, finish; //
    declare start and final time
    start = clock();
    // your code is here, do something and then g
    finish = clock();
    ( (finish - start)/CLOCKS_PER_SEC );
    ...
}
```

## OUR RESULTS

We present our results in Fig. ??.

## CONCLUSIONS AND PERSPECTIVES

What a wonderful world!

- 
- [1] W. Heisenberg, Zeits. f. Physik **77**, 1 (1932).
  - [2] G. A. Miller, A. K. Opper, and E. J. Stephenson, Annu. Rev. Nucl. Sci. **56**, 253 (2006).
  - [3] A. Ekström, G. R. Jansen, K. A. Wendt, G. Hagen, T. Papenbrock, B. D. Carlsson, C. Forssen, M. Hjorth-Jensen, P. Navratil, and W. Nazarewicz, Phys. Rev. C **91**, 051301(R) (2015).
  - [4] B. A. Brown, A. Arima and J. B. McGrory, Nucl. Phys. **A277**, 77 (1977) and references therein.