# FYS 4150 Project 1 Report

Marc Kidwell Pestana

## 1 Abstract

The aim of this project is to get familiar with various vector and matrix operations, from dynamic memory allocation to the usage of programs in the library package of the course.

## 2 Introduction

## 3 Software Design, Theoretical Models, and Algorithms

### 3.1 Software Design

The following list contains a brief description of software design practices and guidelines for how I design, code, and test software programs. These principles were learned either from training courses I took at the Jet Propulsion Laboratory, La Canada California, or are based on personal experience obtained by building software systems for the Laboratory.

- Prototyping: begin with simple functions that simulate the software functional requirements
- l lay the groundwork for my software programs by writing down in my own words, what I want the program to accomplish and how the program will accomplish it. This forms the basis for what are commonly refered to as functional requirments!
- Up front development: do the hard part first. This can mean either getting a clear understanding of the most central and most complicated parts of the program written into the documentation, building the data source needed by the program to function, writing test programs, or any other development that's needed before the program can function or be tested properly.

In the case of Project 1, navigating the class repository and class documentation to be the most challenging, followed by establishing the development platform, overcoming my resistence to using C++ in which I have no fluency, and understanding the details of the algorithm. That is the order in which I approached project 1.
With respect to the development of the C++ code for Project 1:

- Project 1 requires an input parameter determining the number of steps to be used by the algorithm in it's approximation to the solution of the the one-dimensional Poisson equation with Dirichlet boundary conditions. I decided to have the program read the input parameter from the command line.
- Project 1 requires the input of a
- I used this parameter to build the input matrix to the algorithm.
- itemize Implement the algorithm
- itemize establish test of the algorithm.

## 3.2 Theoretical Models for Project 1

### 3.2.1 Project 1a

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:

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

and we define the discretized approximation to uu as vivi with grid points $x_i = ih$ in the interval from $x_0 = 0tox_{n+1} = 1$. The step length or spacing is defined as $h = \frac{1}{(n+1)}$. We have then the boundary conditions $v_0 = v_{n+1} = 0$. We approximate the second derivative of $u$ with

$-(v_{i+1} + v_{i-1} - 2v_i)/h^2 = f_i$ for $i = 1, ..., n$,

where $f_i = f(x_i)$.
These equations can be re-written as a set of n linear equations in n unknows by distributing the $-1$ and multiplying both sides by $h^2$ and rearranging the terms, which gives

$-v_{i-1} + 2v_i - v_{i+1} = h^2 f_i$ for $i = 1, ..., n$,

Expanding these equations and applying the boundary conditions yields

$$\begin{aligned} 2v_1 - v_2 &= h^2 f_1 \\ -v_1 + 2v_2 - v_3 &= h^2 f_2 \\ -v_2 + 2v_3 - v_4 &= h^3 f_2 \\ &. \\ &. \\ &. \\ -v_{n-1} + 2v_n &= h^3 f_n \end{aligned}$$

Letting $\vec{v}$ the vector of unknowns as follows

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ . \\ . \\ v_n \end{bmatrix}$$

Letting $\vec{b}$ the vector of values for $h^2 f(x)$ evaluated at each $x_i$ as follows

$$\vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_n \end{bmatrix} = \begin{bmatrix} h^2 f(x_1) \\ h^2 f(x_2) \\ . \\ . \\ h^2 f(x_n) \end{bmatrix}$$

So the following matrix equation holds

$$\begin{bmatrix} 2 & -1 & .. & .. & .. & .. & .. & .. \\ -1 & 2 & -1 & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & -1 & 2 & -1 \\ .. & .. & .. & .. & .. & .. & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ . \\ . \\ v_n \end{bmatrix} = \begin{bmatrix} h^2 f(x_1) \\ h^2 f(x_2) \\ . \\ . \\ h^2 f(x_n) \end{bmatrix}$$

$$Let \quad \hat{A} = \begin{bmatrix} 2 & -1 & .. & .. & .. & .. & .. & .. \\ -1 & 2 & -1 & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & -1 & 2 & -1 \\ .. & .. & .. & .. & .. & .. & -1 & 2 \end{bmatrix} \quad then,$$

$\hat{A}\vec{v} = \vec{b}$

#### 3.2.2 Project 1b

I will now develop an algorithm to solve a generalization of the "tridiagonal" system introduced in section 1a. A tridiagonal matrix is a special form of banded matrix where all the elements are zero except for those on and immediately above and below the leading diagonal. The above tridiagonal system can be generalized into the following system:

$a_i u_{i-1} + b_i u_i + c_{i+1} = f_i$

In order that Gaussian Elimination is guaranted to yeild a solution to the tridiagonal system if the elements of $\hat{A}$ if the upper diagonal elements $a_n$, the diagonal elements $d_n$, and the lower diagonal elements $c_i j$ statisfy the following relations

$|b_1| > |c_1|, \quad and \quad , \quad |b_n| > |a_n| \quad and \quad |b_n| \leq |a_n| + |c_n|$

## 4 Results and discussion

## 5 Conclusions and perspectives

## 6 Appendix with extra material

## 7 Bibliography