Introduction

This report is going to look in to matrix handling using different approaches. Comparing LU decomposition and tridiagonal decomposition approach. An attempt on c++ programming is done to look at the differences in methods in run time and accuracy.

Theory

Poisson's equation:

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

Which becomes in one dimension with symmetric Φ

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

Starting by looking at the relation $\mathbf{Av} = \tilde{\mathbf{b}}$

Where $\mathbf{A} =$

$$\begin{bmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \vdots \\ 0 & -1 & 2 & -1 & 0 & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \dots & \dots & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 2 \end{bmatrix}$$

and $\mathbf{v} =$

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \dots \\ v_{n-1} \\ v_n \end{bmatrix}$$

$$f_i = -\frac{v_{i+1} + v_{i-1} - 2v_i}{h^2} \tag{3}$$

From 3 can be rewritten as a linear equation $\mathbf{Av} = \mathbf{b}$ where $\mathbf{b} = h^2\mathbf{f}$. This by

$\mathbf{b} = -h^2 * \frac{v_{i+1}+v_{i-1}-2v_i}{h^2}$

$\mathbf{b} = a * v_{i+1} + b * v_i + c * v_{i-1}$

Where a = c = -1 and b = 2.

Which equals $A(i) * v(i)$.

The function $f(i) = e^{-10x}$.

The relative error is given by

$$\epsilon_i = \log_{10}(|\frac{v_i - u_i}{u_i}|) \tag{4}$$

Computation start with a forward substitution the a backward substitution to get the approximation of the value v in the equation $\mathbf{Av} = \mathbf{b}$. The important bit of code that gives the first method:

```
{double btemp = b(0);
u(1) = f(0)/btemp;
for(i=1 ; i < n ; i++) {
     temp(i) = c(i-1)/btemp;
     btemp = b(i)-a(i)*temp(i);
     u(i) = (f(i) - a(i)*u(i-1)/btemp);
     }

//cout<<u<<endl;
//backward substitution loop
for(i=n-2; i>=0; i--){
     u(i) -= temp(i+1)*u(i+1);
}}
```

Codes:

plot.py , main.cpp

Found in Project1/project1 at `https://github.com/magnusji/compfys.git`

Results

Table 1: Run time for each method

N	Tri	LU
10	5.5e-05	0.000175
100	0.000151	0.000858
1000	0.00101	0.07432

The Table 1 shows the difference in runtime between the two methods. And Figure 1 shows how the methods compare against closed form solution.

The relative error is given by Equation 4 and shows $epsu = infepsv = inf$.

More results in `https://github.com/magnusji/compfys.git`.

Discussion

The run time for the first method is always better than for LU decomposition. And also didn't have that much problems with larger matrices. The LU decomposition couldn't go further than size 1000X1000 matrix.

But from Figure 1 it seems like the LU decomposition method follows the curve better than the first method. But since the relative error maximum only shows inf it is hard to say which is more accurate.

There is much in the programs that should be improved. Both the relative error should be improved so that it is working, the max error is larger than it looks like in the printed file. Also the plot should be done by reading from file instead of what is done here by copying values. So much to be improved.

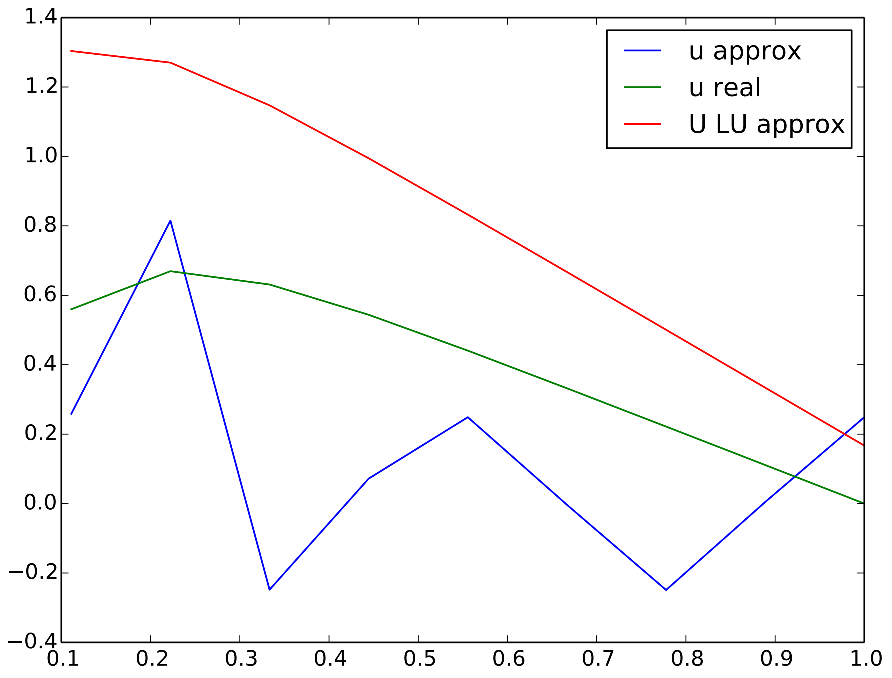

Figure 1: Approximations for Tridiagonal system and LU decomposition and the exact soulution