FYS3150 - Project 1

maziark

September 2017

Link to project

1 Project 1 - A

Given in the project text : $f(x_i) represented as f_i$

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

This fraction can be rewritten as :

$$-v_{i-1} + 2v_i - v_{i+1}) = f_i h^2$$

Then if we define a vector of elements v_i :

$$v = \begin{bmatrix} v_0 \\ v_1 \\ . \\ . \\ . \\ v_{n+1} \end{bmatrix}$$

And $b = f_i h^2$ cane be rewritten as a matrix of shape :

$$b = \begin{bmatrix} f_0 h^2 \\ f_1 h^2 \\ . \\ . \\ . \\ f_{n+1} h^2 \end{bmatrix}$$

And A is just a 2D matrix that has the coefficients which can be represented as in :

$$A = \begin{bmatrix} 2 & -1 & 0 & ... & ... & 0 \\ -1 & 2 & -1 & ... & ... & 0 \\ 0 & -1 & 2 & ... & ... & . \\ . & ... & ... & ... & ... & . \\ . & ... & ... & ... & ... & . \\ 0 & 0 & 0 & ... & -1 & 2 \end{bmatrix}$$

Therefore solving the linear equation Av = b; for v gives us the values for the differential equation.
$f(x) = 100e^{-10x}$
$u(x) = 1 - (1 - e^{-10})x - e^{-10x}$

getting the second derivative of u we have :

$$\frac{du}{dx} = (1 - e^{-10}) - (-10)e^{-10x}$$

$$\frac{d^2u}{du^2} = -(-10)(-10)e^{-10x} = -100e^{-10x}$$

And so $\frac{d^2u}{du^2} = -f(x)$.

2 Project 1 - B

As requested in the assignment's text I made three vectors called : a, b, c to represent the non-zero values of tridiagonal matrix.

f_i is the value of $f(x_i)$. And similarly u_i is defined as $u(x_i)$.

The algorithm to simplify the Tridiagonal matrix is rather simple, only a two step process. So will result in a linear processing time. (O (n)).

In the first step we eliminate the coeficient of a_i and update the value of b, and f: (I've used b' for the new value of b, and f' for the updated value of f)

to update value of b_i

$$b'_i = b_i - \frac{a_i * c_{i-1}}{b'_{i-1}}$$

Now knowing that there is no element a, for the first row :

$b'_i = b_i, and f'_i = f_i$ to update f_i :

$$f'_i = f_i - \frac{a_i * f'_{i-1}}{b'_{i-1}}$$

Now our matrix A is shaped like :

$$A = \begin{bmatrix} b'_1 & c_1 & 0 & ... & ... & 0 \\ 0 & b'_2 & c_2 & ... & ... & 0 \\ 0 & 0 & b' & ... & ... & . \\ . & ... & ... & ... & ... & . \\ . & ... & ... & ... & ... & . \\ 0 & 0 & 0 & ... & 0 & b'_n \end{bmatrix}$$

And therefore, we can find the value of u_n with just a simple division :

$$u_n = f'_n / b'_n$$

And knowing that value, we can find the rest with a backward substitution :

$$u_i = \frac{f'_i - c_i * u_{i+1}}{b_i}$$

Now no matter the size of the array, problem can be solved in a linear method. with 3n floating point operations.

3 Project 1 - C

Following the solution of part B, and knowing that in this particular project the value of a, b, and c are known (a = c = -1 and b = 2). We can simplify it a bit further :

Forward substitution :

$$b'_i = 2 - \frac{-1 * -1}{b'_{i-1}}$$

which can be rewritten in :

$$b'_i = 2 - \frac{1}{b'_{i-1}} = \frac{i+1}{i}$$

This equation can simply be proved with mathematical induction :

assuming the base case be n = 1 :

$b_n = \frac{n+1}{n} = \frac{1+1}{1} = 2$ Which is true;

Now I assume equation can be true for the case n = k; so : $b_k = \frac{k+1}{k}$ and now for n = k+1 we have :

$b_{k+1} = 2 - \frac{1}{b_k} = 2 - \frac{k}{k+1} = \frac{2(k+1)-k}{k+1} = \frac{2k+2-k}{k+1} = \frac{k+2}{k+1}$

Which shows that we can simplify $b'_i = \frac{i+1}{i}$.

And now I try to simplify the equation for finding the value of f'_i

$$f'_i = f_i - \frac{a_i * f'_{i-1}}{b'_{i-1}} = f_i - \frac{-1 * f'_{i-1}}{\frac{i}{i-1}} = f_i + \frac{(i-1)f'_{i-1}}{i}$$

Table 1: runtime

10^N	Runtime (Matrix)
1	0.000037
2	0.000023
3	0.000170
4	0.001285
5	0.013731
6	0.136821
7	1.404342

4 Project 1 - D

Table 2: Comparing results : Optimized V.S. LU

N	Runtime (Matrix)	Runtime (LU)	RelativeError (Matrix)	RelativeError (LU)
10	0.000038	0.000439	0.301744	0.000000
100	0.000021	0.007992	0.037488	0.000000
1000	0.000153	1.126730	0.003849	0.000000

(This table is the outcome of the program project1.cpp). Since the LU decomposition should be of $O(n^3)$ which is two much processing for a simple matrix. And therefore it will be impossible to use it for $N = 10^6$.

And in comparison the algorithm that simplifies the matrix, considering that it is a Tridiagonal matrix, will only grow in O(n), and also requires a memory of size n. And it can be seen that the relative error getting smaller proportionately.

The function that measures the CPU runtime goes as :

```
TYPE CPU_runtime (void (*calculate) (int, TYPE *, TYPE *),
    int N, TYPE *x, TYPE *u, string method_name){
        clock_t start, finish;
        start = clock();
        calculate (N, x, u);
        finish = clock ();

        return ( ((finish - start)*1.0/CLOCKS_PER_SEC));
}
```

This function gets another function as an argument and runs the function, and return the time consumed by the function.