Table of Contents

[INTRODUCTION 2](#_Toc449261093)

[The Functions 3](#_Toc449261094)

[Sfactors Function 3](#_Toc449261095)

[Question 4 4](#_Toc449261096)

[Appendix 5](#_Toc449261097)

[Representative Code 5](#_Toc449261098)

[SFactors, FastSN, FastTN, FastUN 5](#_Toc449261099)

[Matrix Functions – Needed for Main functions after this 5](#_Toc449261100)

[Q2 – Demonstrating FastSN works 5](#_Toc449261101)

[Q4 – Timing DirectSN vs FastSN 5](#_Toc449261102)

[Q5 – Timing Poisson2D using FastSN 5](#_Toc449261103)

All work done in this project is my own unless stated otherwise

INTRODUCTION

This project deals with computing linear systems of equations, .

In the first section I implement functions that's solve the equations of the above form for given matrices . Namely, . N can be of the form , and

In the second section I analyse the speed and efficiency of my written code through calculations of the theoretical maximum speed and the achieved speed.

In the final section, I solve a certain case of Poisson’s equation for the 2D case using my newly created functions and compare it to the results produced in the previous exercise.

Finally, I’d like to mention that the mastery component of this exercise has been incorporated into the various sections before and thus does not have its own dedicated section. I have however highlighted mastery material wherever relevant.

The Functions

In this section I briefly talk about the functions implemented as described by the exercise sheet[[1]](#footnote-1).

# Sfactors Function

Prototype:

**double** **\***SFactors**(int** N**);**

This function takes an integer, N, as input and returns a vector of size with values depending on if 5, 3, or, 2 are a factor of N.

/mastery

If 5 is a factor,

mastery/

If 3 is a factor,

If 2 is a factor,

# FastSN

Prototype:

**int** FastSN**(double** **\***x**,** **double** **\***y**,** **double** **\***w**,** **double** **\***S**,int** N**,** **int** skip**);**

The function checks if N==2, N==3, or N==5 and explicitly calculates the values of for for these cases by minimizing the number of floating point operations through grouping and creating temporary variables to store values of operations needed more than once. If N>2 but Nmod2==0 then FastSN calls FastSN(N/2) and FastTN(N/2). It also performs an additional N-2 operations to calculate the values to be send to each of these functions.

# FastTN

Prototype:

**int** FastTN**(double** **\***x**,** **double** **\***y**,** **double** **\***w**,** **double** **\***S**,int** N**,** **int** skip**);**

Similar to FastSN, FastTN checks if N==1, N==2, N==3, or N==5 and explicitly calculates the values of for for these cases by minimizing the number of floating point operations. If N>2 but Nmod2==0 then FastTN calls FastTN(N/2) and FastUN(N/2) and performs an additional N operations to calculate the result.

# FastUN

Prototype:

**int** FastUN**(double** **\***x**,** **double** **\***y**,** **double** **\***w**,** **double** **\***S**,int** N**,** **int** skip**);**

Similar to FastSN and FastTN, FastUN checks if N==1, N==2, N==3, or N==5 and explicitly calculates the values of for for these cases by minimizing the number of floating point operations. If N>2 but Nmod2==0 then FastUN calls FastTN(N/2) twice and performs an additional 4\*N-2 operations to prepare the values and calculate the result.

# Others

Functions for directly computing the matrices, Sn, Tn, and Un are in the matrix\_functions.c file at the end.

There is also a main function titled “Q2 – demonstrating FastSN works” that does exactly as the title says. For an inputted value of N it first computes the using direct matrix multiplication such that for as we know that . It then uses FastSN to demonstrate that we get back the desired .For invalid inputs of N, it can be seen that FastSN returns -1.

A couple of examples of the output have been produced below:

/mastery

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 12:23:46

Date: Apr 24 2016

Enter N: 5

Computed y: direct matrix multiplication by Sn\*2/N

1 | 3.07768

2 | -1.37638

3 | 0.726543

4 | -0.32492

Returned Value: 0

1 | 1

2 | 2

3 | 3

4 | 4

mastery/

Enter N: 16

Computed y: direct matrix multiplication by Sn\*2/N

1 | 10.1532

2 | -5.02734

3 | 3.29656

4 | -2.41421

5 | 1.87087

6 | -1.49661

7 | 1.2185

8 | -1

9 | 0.820679

10 | -0.668179

11 | 0.534511

12 | -0.414214

13 | 0.303347

14 | -0.198912

15 | 0.0984914

Returned Value: 0

1 | 1

2 | 2

3 | 3

4 | 4

5 | 5

6 | 6

7 | 7

8 | 8

9 | 9

10 | 10

11 | 11

12 | 12

13 | 13

14 | 14

15 | 15

Speed and Efficiency – Question 4

I start by comparing the speed of using FastSN to a direct matrix multiplication function that I refer to as the DirectSN method.

# DirectSN vs FastSN

## Time Taken

Below are the tables for the time taken by FastSN compared to the time taken by the Direct Matrix Multiply for the cases of , and .

TABLE 1

TABLE 2

TABLE 3

For small values of N, that is when N<10, the timing program is not accurate enough to distinguish between DirectSN and FastSN.

**FastSN is twice as fast as DirectSN for .**

**FastSN is 10x faster than DirectSN for .**

FastSN is faster than DirectSN because it requires far fewer floating point operations. Tables comparing the number of operations needed for DirectSN compared to FastSN can be found in the appendix.

## Memory Comaprison

Furthermore, it is worth mentioning that, although memory allocation has not been timed in the above examples, the memory requirements for the two methods are very different and thus can also lead to performance increases in using FastSn for large N.

Storing the matrix grows on an order of and becomes too large to store on the RAM after – approximately – when . The size of the matrix is

which has a space requirement of

Table 1: Size of Sn matrix for powers of 2 [[[2]](#footnote-2)]

|  |  |
| --- | --- |
| Power of 2 | Size of Sn Matrix (gb) |
| 13 | 0.499877937 |
| 14 | 1.999755867 |
| 15 | 7.999511726 |
| 16 | 31.99902344 |
| 17 | 127.9980469 |

In comparison the memory requirement for FastSN – in gb - is the space required for SFactors (size N/2) and the matrix w needed for workings (size N-1)

This is of the order and clearly grows much more slowly and thus we can apply it to much larger vectors . The table below gives some numbers for the same range as the one for DirectSN and shows how it requires significantly less space.

Table 2: Size of memory needed for solving FastSN for powers of 2

|  |  |  |
| --- | --- | --- |
| **Power** | **N** | **Size of Storeage needed (gb)** |
| 13 | 8192 | 9.15453E-05 |
| 14 | 16384 | 0.000183098 |
| 15 | 32768 | 0.000366203 |
| 16 | 65536 | 0.000732414 |
| 17 | 131072 | 0.001464836 |

Now before examining the theoretical maximum speed, I discuss the number of floating point operations FastSN takes.

# Number of Operations

Number of operations needed to do when we try to minimize operations would be 0 as

Number of operations needed to do when we try to minimize operations would be 3 as

Here we save one floating point operation by storing the value of in a temporary variable and using it for both and . Also multiplication to 1 is not necessary.

Similarly, the values for calculating the minimized number of operations for have been calculated and put in the table below:

Table 3: Minimized number of +- and \* operations needed

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | FastSN +- | FastSN \* | FastSN Total | FastTN +- | FastTN \* | FastTN Total | FastUN +- | FastUN \* | FastUN Total |
| 2 | 0 | 0 | 0 | 2 | 1 | 3 | 2 | 4 | 6 |
| 3 | 2 | 2 | 4 | 4 | 2 | 6 | 6 | 6 | 12 |
| 5 | 8 | 8 | 16 | 12 | 8 | 20 | 20 | 18 | 38 |

As expressed in the first section, the total number of operations FastSN, FastTn, and FastUN take depends recursively on themselves for N/2 and then N/4 and so on and so forth until N==2, or N==3 or N==5. Look back for formulae.

The formulae for number of additions and subtractions is given below for all three functions:

And the formulae for number of multiplications is:

The tables – for 2, 3, and 5 - in the following pages give the number of +/- and \* needed for each value of N. Finding the values that have been fitted for the given equation

can be found in the appendix. It has been done in Maple and MATLAB and both results agree to over 8 decimal places.

Table 4: Number of Operations Required case 2 times powers of 2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Power | N | W\_S2N +- | W\_S2N \* | W\_S2N | W\_T2N +- | W\_T2N \* | W\_T2N | W\_U2N +- | W\_U2N \* | W\_U2N |
| 1 | 2 | 0 | 0 | 0 | 2 | 1 | 3 | 2 | 4 | 6 |
| 2 | 4 | 4 | 1 | 5 | 8 | 5 | 13 | 10 | 10 | 20 |
| 3 | 8 | 18 | 6 | 24 | 26 | 15 | 41 | 30 | 26 | 56 |
| 4 | 16 | 58 | 21 | 79 | 72 | 41 | 113 | 82 | 62 | 144 |
| 5 | 32 | 160 | 62 | 222 | 186 | 103 | 289 | 206 | 146 | 352 |
| 6 | 64 | 408 | 165 | 573 | 456 | 249 | 705 | 498 | 334 | 832 |
| 7 | 128 | 990 | 414 | 1404 | 1082 | 583 | 1665 | 1166 | 754 | 1920 |
| 8 | 256 | 2326 | 997 | 3323 | 2504 | 1337 | 3841 | 2674 | 1678 | 4352 |
| 9 | 512 | 5340 | 2334 | 7674 | 5690 | 3015 | 8705 | 6030 | 3698 | 9728 |
| 10 | 1024 | 12052 | 5349 | 17401 | 12744 | 6713 | 19457 | 13426 | 8078 | 21504 |
| 11 | 2048 | 26842 | 12062 | 38904 | 28218 | 14791 | 43009 | 29582 | 17522 | 47104 |
| 12 | 4096 | 59154 | 26853 | 86007 | 61896 | 32313 | 94209 | 64626 | 37774 | 102400 |
| 13 | 8192 | 129240 | 59166 | 188406 | 134714 | 70087 | 204801 | 140174 | 81010 | 221184 |
| 14 | 16384 | 280336 | 129253 | 409589 | 291272 | 151097 | 442369 | 302194 | 172942 | 475136 |
| 15 | 32768 | 604374 | 280350 | 884724 | 626234 | 324039 | 950273 | 648078 | 367730 | 1015808 |
| 16 | 65536 | 1296142 | 604389 | 1900531 | 1339848 | 691769 | 2031617 | 1383538 | 779150 | 2162688 |
| 17 | 131072 | 2767060 | 1296158 | 4063218 | 2854458 | 1470919 | 4325377 | 2941838 | 1645682 | 4587520 |
| 18 | 262144 | 5883660 | 2767077 | 8650737 | 6058440 | 3116601 | 9175041 | 6233202 | 3466126 | 9699328 |
| 19 | 524288 | 12466386 | 5883678 | 18350064 | 12815930 | 6582727 | 19398657 | 13165454 | 7281778 | 20447232 |
| 20 | 1048576 | 26330890 | 12466405 | 38797295 | 27029960 | 13864505 | 40894465 | 27729010 | 15262606 | 42991616 |

Fitting W\_S2N +-, W\_S2N \*, W\_S2N to gives

|  |  |  |  |
| --- | --- | --- | --- |
| A2 +/- | 1.33336449 | B2 +/- | -1.55619325 |
| A2 \* | 0.66666341 | B2 \* | -1.44437825 |
| A2 | 2.00002790 | B2 | -3.00057149 |

Table 5: Number of Operations Required case 3 times powers of 2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Power | N | W\_S3N +- | W\_S3N \* | W\_S3N | W\_T3N +- | W\_T3N \* | W\_T3N | W\_U3N +- | W\_U3N \* | W\_U3N |
| 1 | 3 | 2 | 2 | 4 | 4 | 2 | 6 | 6 | 6 | 12 |
| 2 | 6 | 10 | 4 | 14 | 16 | 8 | 24 | 18 | 16 | 34 |
| 3 | 12 | 36 | 12 | 48 | 46 | 24 | 70 | 54 | 40 | 94 |
| 4 | 24 | 104 | 36 | 140 | 124 | 64 | 188 | 138 | 96 | 234 |
| 5 | 48 | 274 | 100 | 374 | 310 | 160 | 470 | 342 | 224 | 566 |
| 6 | 96 | 678 | 260 | 938 | 748 | 384 | 1132 | 810 | 512 | 1322 |
| 7 | 192 | 1616 | 644 | 2260 | 1750 | 896 | 2646 | 1878 | 1152 | 3030 |
| 8 | 384 | 3748 | 1540 | 5288 | 4012 | 2048 | 6060 | 4266 | 2560 | 6826 |
| 9 | 768 | 8526 | 3588 | 12114 | 9046 | 4608 | 13654 | 9558 | 5632 | 15190 |
| 10 | 1536 | 19106 | 8196 | 27302 | 20140 | 10240 | 30380 | 21162 | 12288 | 33450 |
| 11 | 3072 | 42316 | 18436 | 60752 | 44374 | 22528 | 66902 | 46422 | 26624 | 73046 |
| 12 | 6144 | 92832 | 40964 | 133796 | 96940 | 49152 | 146092 | 101034 | 57344 | 158378 |
| 13 | 12288 | 202058 | 90116 | 292174 | 210262 | 106496 | 316758 | 218454 | 122880 | 341334 |
| 14 | 24576 | 436894 | 196612 | 633506 | 453292 | 229376 | 682668 | 469674 | 262144 | 731818 |
| 15 | 49152 | 939336 | 425988 | 1365324 | 972118 | 491520 | 1463638 | 1004886 | 557056 | 1561942 |
| 16 | 98304 | 2009756 | 917508 | 2927264 | 2075308 | 1048576 | 3123884 | 2140842 | 1179648 | 3320490 |
| 17 | 196608 | 4281670 | 1966084 | 6247754 | 4412758 | 2228224 | 6640982 | 4543830 | 2490368 | 7034198 |
| 18 | 393216 | 9087642 | 4194308 | 13281950 | 9349804 | 4718592 | 14068396 | 9611946 | 5242880 | 14854826 |
| 19 | 786432 | 19223876 | 8912900 | 28136776 | 19748182 | 9961472 | 29709654 | 20272470 | 11010048 | 31282518 |
| 20 | 1572864 | 40544920 | 18874372 | 59419292 | 41593516 | 20971520 | 62565036 | 42642090 | 23068672 | 65710762 |

Fitting W\_S3N +-, W\_S3N \*, W\_S3N to gives

|  |  |  |  |
| --- | --- | --- | --- |
| A3 +/- | 1.33335400 | B3 +/- | -1.66927409 |
| A3 \* | 0.66666341 | B3 \* | -1.72318864 |
| A3 | 2.00002790 | B3 | -3.39246273 |

Table 6: Number of Operations Required case 5 times powers of 2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Power | N | W\_S5N +- | W\_S5N \* | W\_S5N | W\_T5N +- | W\_T5N \* | W\_T5N | W\_U5N +- | W\_U5N \* | W\_U5N |
| 1 | 5 | 8 | 8 | 16 | 12 | 8 | 20 | 20 | 18 | 38 |
| 2 | 10 | 28 | 16 | 44 | 42 | 26 | 68 | 42 | 36 | 78 |
| 3 | 20 | 88 | 42 | 130 | 104 | 62 | 166 | 122 | 92 | 214 |
| 4 | 40 | 230 | 104 | 334 | 266 | 154 | 420 | 286 | 204 | 490 |
| 5 | 80 | 574 | 258 | 832 | 632 | 358 | 990 | 690 | 468 | 1158 |
| 6 | 160 | 1364 | 616 | 1980 | 1482 | 826 | 2308 | 1582 | 1036 | 2618 |
| 7 | 320 | 3164 | 1442 | 4606 | 3384 | 1862 | 5246 | 3602 | 2292 | 5894 |
| 8 | 640 | 7186 | 3304 | 10490 | 7626 | 4154 | 11780 | 8046 | 5004 | 13050 |
| 9 | 1280 | 16090 | 7458 | 23548 | 16952 | 9158 | 26110 | 17810 | 10868 | 28678 |
| 10 | 2560 | 35600 | 16616 | 52216 | 37322 | 20026 | 57348 | 39022 | 23436 | 62458 |
| 11 | 5120 | 78040 | 36642 | 114682 | 81464 | 43462 | 124926 | 84882 | 50292 | 135174 |
| 12 | 10240 | 169742 | 80104 | 249846 | 176586 | 93754 | 270340 | 183406 | 107404 | 290810 |
| 13 | 20480 | 366806 | 173858 | 540664 | 380472 | 201158 | 581630 | 394130 | 228468 | 622598 |
| 14 | 40960 | 788236 | 375016 | 1163252 | 815562 | 429626 | 1245188 | 842862 | 484236 | 1327098 |
| 15 | 81920 | 1685716 | 804642 | 2490358 | 1740344 | 913862 | 2654206 | 1794962 | 1023092 | 2818054 |
| 16 | 163840 | 3589898 | 1718504 | 5308402 | 3699146 | 1936954 | 5636100 | 3808366 | 2155404 | 5963770 |
| 17 | 327680 | 7616722 | 3655458 | 11272180 | 7835192 | 4092358 | 11927550 | 8053650 | 4529268 | 12582918 |
| 18 | 655360 | 16107272 | 7747816 | 23855088 | 16544202 | 8621626 | 25165828 | 16981102 | 9495436 | 26476538 |
| 19 | 1310720 | 33962192 | 16369442 | 50331634 | 34836024 | 18117062 | 52953086 | 35709842 | 19864692 | 55574534 |
| 20 | 2621440 | 71419654 | 34486504 | 105906158 | 73167306 | 37981754 | 111149060 | 74914926 | 41477004 | 116391930 |

Fitting W\_S3N +-, W\_S3N \*, W\_S3N to gives

|  |  |  |  |
| --- | --- | --- | --- |
| A5 +/- | 1.33334808 | B5 +/- | -1.18511451 |
| A5 \* | 0.66666222 | B5 \* | -1.05896701 |
| A5 | 2.00001030 | B5 | -2.24408152 |

# Theoretical Maximum Speed

I measure theoretical maximum speed of my function as

Appendix

# Representative Code

Below is all the code for the exercise

## SFactors, FastSN, FastTN, FastUN

## Matrix Functions – Needed for Main functions after this

Note that these functions are not needed for SFactors, FastSN, FastTN, and FastUN.

## Q2 – Demonstrating FastSN works

## 

## Q4 – Timing DirectSN vs FastSN

## Q5 – Timing Poisson2D using FastSN

## Maple Code for finding number of operations

## Maple Code for finding coefficients

##### MATLAB Code for finding coefficients

This gives the same result as the Maple code to 8 decimal places.

1. Dr. Dan Moore, http://wwwf.imperial.ac.uk/~drmii/M3SC\_2016/index.htm [↑](#footnote-ref-1)
2. Yadu Bhageria, Exercise 2, Section: Poisson’s Equation in 1D, Table 1 [↑](#footnote-ref-2)