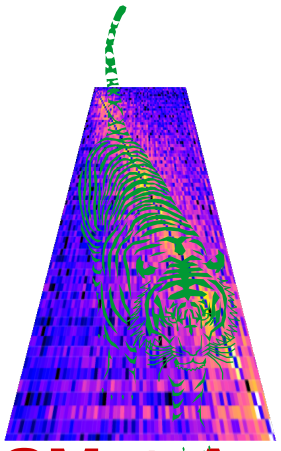
Short Manual of SMatrAn ver. 0.150117

Tatsuya Usuki

March 31, 2015



SMatrAn is a numerical simulator for scattering matrix (S-matrix) derived from wave equation. There are other simulators which can output S-matrix, but they generally have an approach to transform results on the other forms to a part of S-matrix. On the other hand, an accurate S-matrix leads to quantitative evaluation of scattering wave. The S-matrix has the whole of amplitude and phase information for reflected and transmitted waves from a complicated scatterer. SMatrAn focuses on high-accuracy calculation for all elements of S-matrix. SMatrAn can provide useful information to engineers and scientists.

Source files of SMatrAn for electromagnetic waves were opened to the public. Everyone can download and use them for free. Please compile executable files of SMatrAn by one's self, because the user should read source files of at least once. It's as well as Linux, but when you also use MinGW in Windows, this compiler can be installed easily. (Windows command shell is used to install and carry out SMatrAn on Windows. MSYS is unnecessary.) Our web page prepares scripts for compiling executable files from the source files and testing simulation of three-dimensional optical structure. By downloading a whole of directory including source files and others, you can quickly build simulation environment. This manual shows whole process of the SMatrAn.

SMatrAn works on batch processing. We have to use directly input files, because SMatrAn does not provide GUIs of generating input data. This manual shows a typical procedure of building simulation environment after short introduction of SMatrAn. Next it explains how to edit input files to SMatrAn, and it shows how to analyze scattering wave by using output files from SMatrAn while introducing an actual procedure.

You can briefly use SMatrAn after reading this manual. S-matrix provided by SMatrAn will give you detailed analysis on complicated scattering in optical structure. I can hardly shorten its update period, but I will respond to needs from users as much as possible. Please let me know if you have any questions and comments.

Contents

1 Introduction of SMatrAn **4**

1.1 Background of development for SMatrAn 4

1.2 User of SMatrAn, and its purpose 5

1.3 Programming language and environment 6

2 Components of SMatrAn **8**

2.1 Preparation group 8

2.2 Simulation group 9

2.3 Format of input-files 9

3 Construction of SMatrAn **11**

3.1 Compiling executable files 11

3.2 Running an example 12

4 Development schedule and history **15**

5 LICENSE AGREEMENT **16**

1 Introduction of SMatrAn

SMatrAn could numerically calculate S-matrix which shows wave scattering. We will use SMatrAn as an abbreviation for 'Scattering Matrix Analyzer' or 'Scattering Matrix Analysis', because there was not a suitable name for numerical method specialized to S-matrix solver. Figure 1.1 is a logo to be used in WEB pages and presentations. To address

Figure 1.1: Logo of SMatrAn.

general device-designers, we created the above name and logo. We do not intend to restrict them, and then we can spell 'SMATRAN' for an example. SMatrAn is associated with *sumatran*, but we still haven't decided how to pronounce it.

This chapter shows introduction from background of this development to development environment for SMatrAn. Chapter 2 introduces executable files of SMatrAn which opens to the public. Chapter 3 explains compiling SMatrAn and how to simulate an example. Chapter **??** and **??** note how to set up parameters and how to read outputs. Chapter 4 shows schedule of release and history. The person who doesn't have time may begin to read from Chapter 3, but please be sure to read carefully "LICENSE AGREEMENT" in Chapter 5. Please let me know if you have any questions and comments.

1.1 Background of development for SMatrAn

Detailed analysis of S-matrix elements is possible by imposing joint condition between propagated waves in 2D-/3D-medium with complex structure and waveguide modes for both edges of the structure. As a one example, figure 1.2 shows 3D structure which joins waveguides with different shapes. Incident wave from 'Bottom waveguide' (left side in the figure) is scattered by central 'Joint structure' into reflected wave to 'Bottom waveguide' and transmitted wave to 'Top waveguide' (right side in the figure). Each element (complex number) of the S-matrix represents square root of flow-rate for a scattered mode normalized by incident flow-rate, and its phase. Waves as waveguide-modes in Fig. 1.2 are

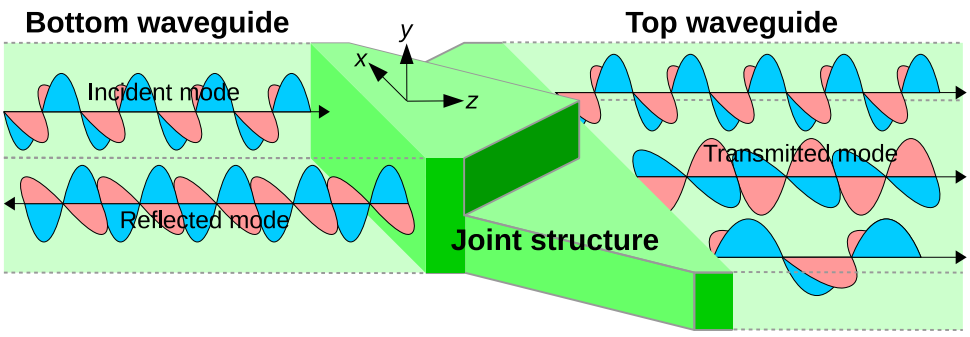


Figure 1.2: Joint structure between two waveguides is shown as a 3D example. The z-axis is oriented from the bottom to the top.

strongly simplified for illustration purpose, but we have to consider every wave-mode including evanescent mode in the actual numerical calculation. Please read references [1, 2, 3] to know exact definition of the S-matrix and details of the numerical method.

SMatrAn does not require any special limitation for wave equations for defining wave propagation. We applied SMatrAn to S-matrix analysis on Shrödinger equation and Maxwell equation. The first use of a method for SMatrAn is numerical calculation of electro-conductibity in 2D-electron system for mesoscopic region[1]. As calculation method of quantum transports, matrix method [4] and recursive Green-function method [5] were well known until then. The former was developed for tunneling phenomena in quantum wells, and it is stable calculation. It, however, requires sequential solving of eigenproblem at each connection in 2D/3D scatterer, and then each step in more complex structure accumulates numerical error due to unperfect orthogonality of eigenvectors. The latter was applied to calculate 2D electron system, especially for complex scattering as Anderson localization [5]. It does not require to solve eigenproblem except for both edge regions of the scatterer, but numerical calculation becomes unstable depending on a condition. The method by ref. [1] resolved the disadvantages of the above methods, and then it can calculate S-matrix stably and accurately. Even now, this method is used because of a quantum transport for mesoscopic systems such as graphene [6, 7, 8].

We open SMatrAn as a fullwave 3D electromagnetic (EM) simulator to the public. Numerical stability and analyzing accuracy resulted from SMatrAn will be widely useful in design verification of optical waveguiding including siliconphotonics and RF devices, and that is a reason why the opened SMatrAn focused on a EM-simulator. Feedback from various persons will improve the quality of the simulator.

1.2 User of SMatrAn, and its purpose

User of the opened SMatrAn is supposed as engineer or scientist who has already used some EM-simulator. For the first time using numerical simulation (especially for work), you should choose a simulator used widely in your field. Proprietary simulator may be valuable for the price, but it is still difficult to analyze some scattering processes by it.

We can recommend you to use SMATRAN if you have no choice but to accurately analyze reflection, mode-transition, reversibility or orthogonality in frequency domain (Fig. 1.3). S-matrix, which is also called as S parameter, is directly useful for analyz-

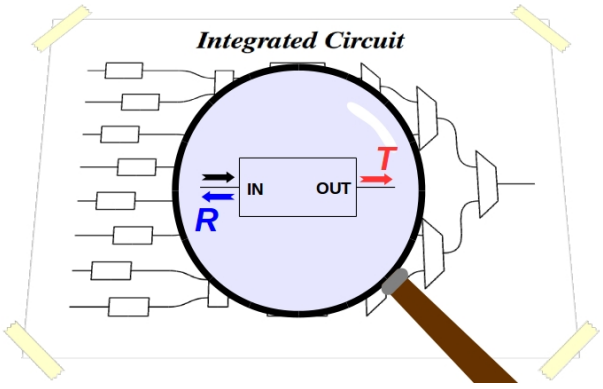


Figure 1.3: SMatrAn is just like a magnifying glass for designing devices.

ing frequency characteristics, because it is often used as an index for optical waveguiding systems and RF transmission lines.

SMatrAn has no approximation except for discretization of wave-equation and numerical error due to floating-point arithmetic. We can discretize the wave-equation with satisfying flow-rate conservation, and then the property of wave-propagation remains after the discretization. Implicit method is applied as stable calculation with converging error. We

also prepare a solver which obtain eigenvectors with higher precision, because eigenvector solver often causes large error. Using the above solution, SMatrAn was designed as satisfying that residual for S-matrix is less than 10^{-5} (usually 10^{-10}).

Please note that its speed will be faster but we give top priority to improving accuracy or evaluation for scattering process. "Fast", "Large" and "Accurate" are justice in the field of scientific computing [9], and then it has top priority to "accuracy" of the scattering phenomena [10]. Nevertheless, SMatrAn partially uses LAPACK for "speed", but it is still not fully adopted with both LAPACK/BLAS and its precision. Also note that a design criterion of SMatrAn is quite opposite to one of FDTD method using explicit method on the purpose of "speed" and "scale".

For build and usage of the SMatrAn, the user needs to have the skills to compile a program from source and to use a shellscript. In no event shall the author be liable for any damages arising in any way out of the use of the SMatrAn. Please read a license in Chapter 5 "LICENSE AGREEMENT."

1.3 Programming language and environment

SMatrAn is written in C99. At 2005, we started to plan electromagetic application based on the previous paper [1]. Waveguide-mode calculating was started at 2010. We are analyzing numerical S-matrix for optical waveguiding from the summer in 2012 [2, 3]. The first version of SMatrAn was written after minimum correction and expansion to the above program sources (from April in 2014).

Mathematic functions and their complex functions for float, double and long double precisions are completely defined by C99 standard, but these are insufficiently defined by C++98/Fortran standard. <tgmath.h> is also useful for writing source code. However, the C99 standard has a disadvantage in that only few compilers support it. We have already used GNU Compiler Correction (GCC) at that time, and then C99 was the best language to build the simulators from scratch. However, as time goes by, C++11 has included all math functions defined by C99, and it implements <ctgmath>. Other ability of C++11 is higher than one of C99 [12]. Visual C++ on Windows supports the C++11 standard, and then we will partially use C++11 for the next update of SMatrAn(Fig. 1.4).

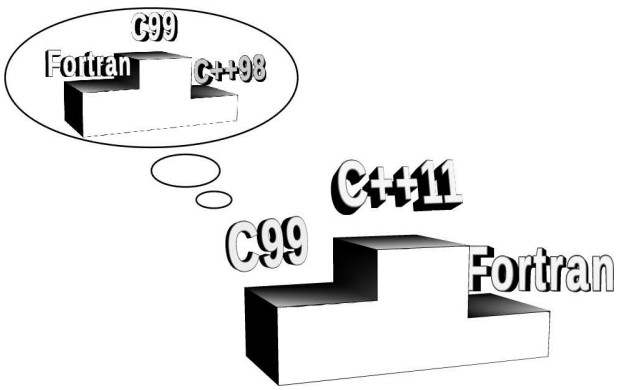


Figure 1.4: The standard of mathematical functions in C++11 is now equal to or better than that in C99.

OS for development is 32-bit Ubuntu 14.04 LTS. Testing environments for SMatrAn are 32-bit and 64-bit Ubuntu 14.04 LTS, and 32-bit and 64-bit Windows 7. As program compiler, we use GCC 4.8.2 or later version. MinGW (GCC version 4.8.1-4 or later) is used as GCC for Widows 7. In other words, SMatrAn was tested for 8 combinations of OS and compiler. These compilers are widely used under GPL or University of Illinois/NCSA License.

CPU for development and one for operation check are shown in table 1.1. SMatrAn

Table 1.1: Two hardwares for this manual.

Architecture	CPU	Clock	Memory	Purpose
32 bit	U2300	1.20 GHz×2	3.9 GiB	Development tool
64 bit	i7-4770	3.40 GHz×8	7.7 GiB	Verification tool

may run on other hardware, because the source code is designed to be independent on it. However, note that eigenvector solver on long double format with the 80-bit floating point (Intel CPU) can give us more accurate results even if we apply the results to other calculation on double precision. The 32 bit hardware of a development environment is the quite low performance present, but it has sufficient speed (about 1/5 of the 64 bit hardware) for trial use.

Next chapter explains details of the simulator.

[Go to table of contents.] [Go to home.]

2 Components of SMatrAn

This chapter introduces the executable files that compose SMatrAn. As shown in Fig. 2.1, 9 executable files are binded into 3 groups. The first group generating 3D parameters of

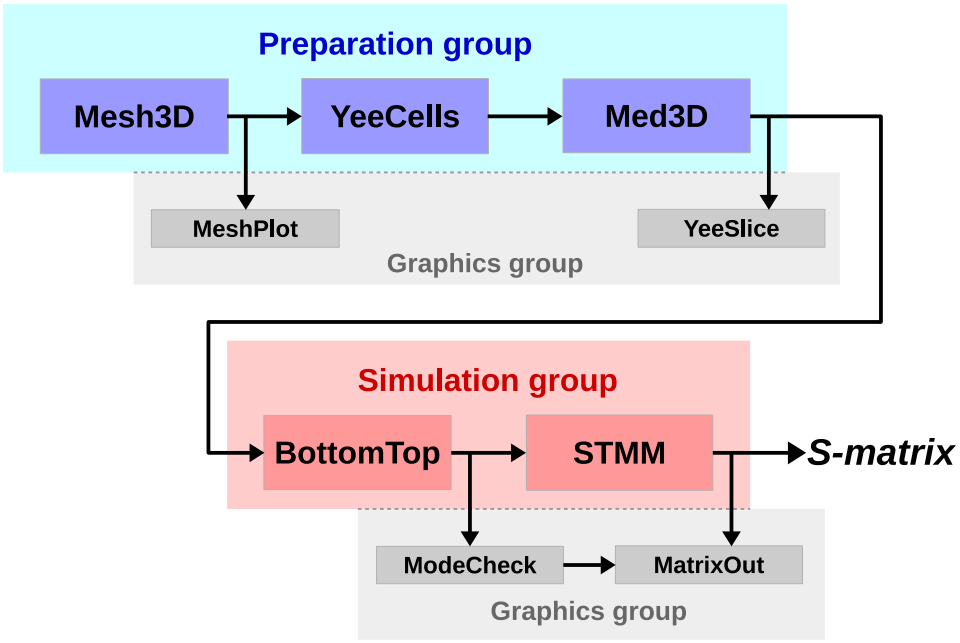


Figure 2.1: Simulation flow from 3D-meshing to S-matrix.

wave-equation is called preparation group. The preparation group is shown as blue color in Fig. 2.1. The next group to calculate S-matrix for the wave-equation is called simulation group as shown in red color The group checking output files in the above is also provided as graphics group in gray color.

2.1 Preparation group

The preparation group consists of 3 executable files as follows.

1. Exec. file name: `Mesh3D`
   - This file generates nonuniform mesh for Yee's lattice.
2. Exec. file name: `YeeCells`
   - This file calculates optical structure in each Yee's cell.
3. Exec. file name: `Med3D`
   - This file calculates electric permittivity and magnetic permeability in each Yee's cell.

After running the above files sequentially, we can then obtain all 3D data required to run the simulation group. We can hardly confirm 3D structure by only using the generated numerical data. The two executable files are then used to visualize the structure of the data.

1. Exec. file name: `MeshPlot`
    - This file for checking Yee's lattice.
2. Exec. file name: `YeeSlice`
    - Slicing operation for checking 3D permittivity distribution.

After confirming the 3D mesh and the spatial distribution of permittivity, we can operate the simulation group.

2.2 Simulation group

The simulation group for computing the wavefunction consists of 3 executable files as follows.

1. Exec. file name: `BottomTop`
    - This file calculates waveguide/evanescent-modes for both the bottom and top waveguides.
2. Exec. file name: `STMM`
    - This file calculates S-matrix by the transfer matrix method.

By running the two executable files, we can obtain the S-matrix. The `STMM` can also output the results of unitarity and other properties of the S-matrix. The following two tools in the graphics group can visualize power distribution of the waveguide modes and residual distribution of matrix elements.

1. Exec. file name: `ModeCheck`
    - This file for checking orthogonality and generating GNUplot data-files for the waveguide/evanescent modes.
2. Exec. file name: `MatrixOut`
    - This file maps matrix elements.

To operate SMatrAn, input files into each executable file require control-file by user and/or data-file by other executable file. Either input-file is ASCII readable text. Next shows format of control files which users should prepare.

2.3 Format of input-files

SMatrAn requires several control files which users have to prepare. Character's length with spaces should be less than 1024 at each line in the files which cannot use multibyte characters

The executable files read numeric characters as a numerical value just after the character '=' just after any character, but spaces are ignored. Here we show examples:

True `L = 22, K_u = 4`

True `L=22K_u=4`

False `=22,K_u=4`

If a numerical value require a mathematical unit or a physical unit, we can use the following units: '`deg`' as °, '`rad`', '`km`', '`m`', '`cm`', '`mm`', '`micron`', '`um`', '`nm`', '`K`', '`degC`' as °C and '`degF`' as °F. The executable files also recognize a space character after the unit as separation from the next numerical value.

True `x_min = -1.5 micron x_max = 1.5 um`

True `x_min = -1.5micron, x_max=1.5 um`

False `x_min = -1.5micron,x_max= 1.5 um`

The above protocol is necessary before constructing SMatrAn.
The next section will show a way how to construct SMatrAn by using an example case.

[Go to table of contents.] [Go to home.]

3 Construction of SMatrAn

This section introduces an example of running an exmaple. We might be saying the same thing again, "IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DAMAGES ARISING IN ANY WAY OUT OF THE USE OF THE SMATRAN WITH/WITHOUT THE EXAMPLE." All files except for the pdf form of this manual are ASCII text. After the user read files and "LICENSE AGREEMENT" in Chapter 5 and understood the contents sufficiently, please use it at your own risk. SMatrAn has been developed by using a combination of Ubuntu and GCC as it was shown in "Programming language and environment." If you use other combinations, please rewrite shell scripts as the need arises. The others will be not explained in the following, but the WEB page also prepares shell scripts in Clang and batch file for Windows. Batch file runs with clicking a mouse.

3.1 Compiling executable files

The example uses directory structure shown on Fig. 3.1. Initial size of the directory is 397 kB except for pdf manuals, but it is increased to 81.7 MB after simulating the example. (When running it on the other structure, it's necessary to change several control files and

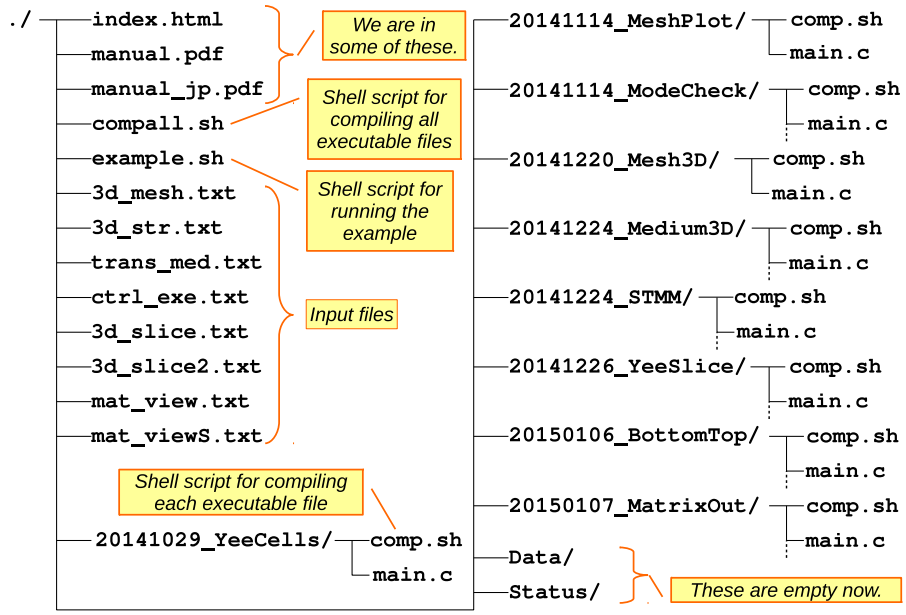


Figure 3.1: An example of initial files and directories. '`./Data/`' and '`./Status/`' are empty before running SMatrAn.

shell scripts.) SMatrAn consists of 9 executable files. The directory with this manual (in Fig. 3.1) has 9 sub directories which correspond to executable files.

It's possible to download all files shown in Fig. 3.1. Each file is small, and then its compiling doesn't use 'make' and carries out by using the gcc command in shell script. (When using batch file in Windows OS, the compiling requires only MinGW or MinGW + Clang without MSYS.) Each sub directory has name with renewal date, and so downloaded files do not require the time-stamp management. For example, the subdirectory `./20141220_Mesh3D` has shell script `comp.sh` in addition to C99 source files and the header file. `comp.sh` is as follows.

```
#!/bin/sh
gcc -Wall -O3 *.c -std=C11 -lm -o ../Mesh3D 2> error_gcc.txt &&
    echo 'compiled Mesh3D'
```

Please change the shebang and compile options as the need arises. Then, run a terminal and move to `./20141220_Mesh3D`. Before running shell, please be sure to check the contents of `comp.sh`.

```
~/SMatrAn/20141220_Mesh3D$ sh comp.sh
```

(The above is an indication when sub directories are built in a directory `SourceFiles` under the home directory `SMatrAn`.)

An empty `error_gcc.txt` is generated in `./20141220_Mesh3D` when the compiling succeeds. When it is not empty, please correct compile options and source files with error/warning message. Other executable files can also be compiled by the same way, but `./compall.sh` is convenient if all 9 executable files are compiled first. `BottomTop` can use '`dgeev`' of LAPACK for higher speed computing. When you now have LAPACK and need high speed, please replace a file with `.lapack` in a file name end with a file with `.c`, and then compile the new file. Other executable file will use LAPACK/BLAS, but the update has a long time. Install of LAPACK/BLAS is recommended for requiring higher speed computing.

3.2 Running an example

Simulating an example is available after compiling all executable files. The directories for compiling them will not be used and can be deleted. Figure 3.2 shows files and directories just before simulating, please check your ones.

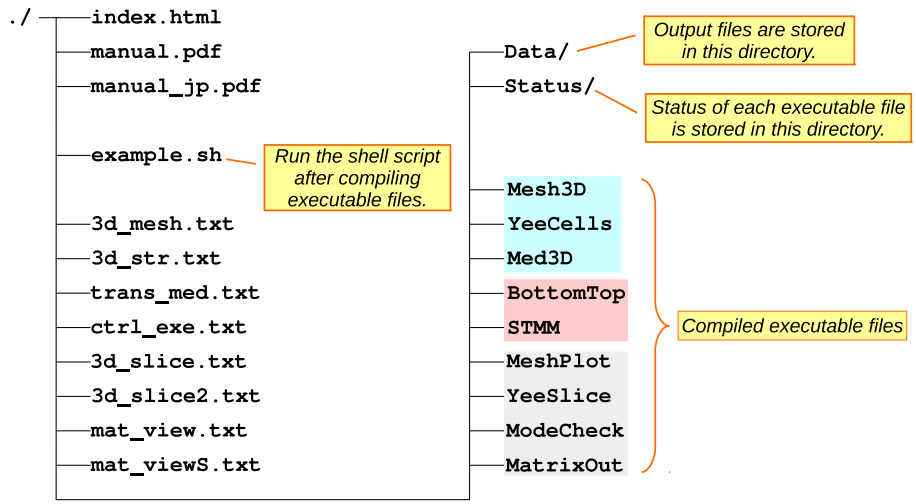


Figure 3.2: Files and directories after compiling all executable files and removing unnecessary directories.

We will use a sell-script `example.sh` for executing 9 files in order. (For running the example for Windows, leave a batch-file `./example.bat` instead of `./example.sh`.) Content of `./example.sh` is shown as follows.

```
#!/bin/sh
./Mesh3D 3d_mesh.txt ./Data/3d_mesh.dat 2> ./Status/Mesh3D.log && \
echo 'Mesh3D finished' && \
./MeshPlot ./Data/3d_mesh.dat 2> ./Status/MeshPlot.log && \
echo 'MeshPlot finished' && \
```

```
./YeeCells ./Data/3d_mesh.dat 3d_str.txt ctrl_exe.txt 2> ./Status/
   YeeCells.log && \
echo 'YeeCells finished' && \
./Med3D trans_med.txt ctrl_exe.txt 2> ./Status/Med3D.log && \
echo 'Med3D finished' && \
./YeeSlice ./Data/3d_mesh.dat 3d_slice.txt 2> ./Status/YeeSlice.log
    && \
echo 'YeeSlice finished' && \
./YeeSlice ./Data/3d_mesh.dat 3d_slice2.txt 2> ./Status/YeeSlice.
   log && \
echo 'YeeSlice finished' && \
./BottomTop ./Data/3d_mesh.dat ctrl_exe.txt 2> ./Status/BottomTop.
   log && \
echo 'BottomTop finished' && \
./ModeCheck ./Data/3d_mesh.dat ctrl_exe.txt 2> ./Status/ModeCheck.
   log && \
echo 'ModeCheck finished' && \
./MatrixOut ./Data/joint_Bottom_check.dat mat_view.txt 2> ./Status/
   MatrixOut_B.log && \
echo 'MatrixOut finished' && \
./MatrixOut ./Data/joint_Top_check.dat mat_view.txt 2> ./Status/
   MatrixOut_T.log && \
echo 'MatrixOut finished' && \
./STMM ./Data/3d_mesh.dat ctrl_exe.txt 2> ./Status/STMM.log && \
echo 'STMM finished'
./MatrixOut ./Data/joint_final.dat mat_viewS.txt 2> ./Status/
   MatrixOut_S.log && \
echo 'MatrixOut finished' && \
```

Please confirm the above content and go to run the example.

```
~/SMatrAn$ sh example.sh
```

Each step of the simulation has been completed in the example process., a terminal shows an end message. When all steps finish, it will wait for new input as follows. (Terminal will close after all steps for Windows OS.)

```
~/SMatrAn$ sh example.sh
Mesh3D finished
MeshPlot finished
YeeCells finished
Med3D finished
YeeSlice finished
YeeSlice finished
BottomTop finished
ModeCheck finished
MatrixOut finished
MatrixOut finished
STMM finished
MatrixOut finished
~/SMatrAn$
```

The precision of numerical results and computing time in the environment of the author are shown on the table 3.1. The one which is Ubuntu more than Windows 7 in the reach of this combination, and Clang tends to be quick at calculation speed more than GCC. A program has been improved so that the calculation precision isn't affected by the environment, but there seems to be little dependence of the OS in compiling by Clang. Therefore, both operating systems will suggest use of Clang to compiling.

Table 3.1: Compiler dependence of numerical calculation on several OSes. $\|\boldsymbol{S}^\dagger\boldsymbol{S}-\boldsymbol{I}\|_{max}$ shows an index as precision of S-matrix. Note that $\|\boldsymbol{A}\|_{max}$ is the max norm defined as $\|\boldsymbol{A}\|_{max}\triangleq max\,|a_{ij}|$. Specifications of 64 bit and 32 bit hardwares are shown in Table 1.1.

OS	item	GCC	Clang
Ubuntu 64 bit	$\|\boldsymbol{S}^\dagger\boldsymbol{S}-\boldsymbol{I}\|_{max}$	1.88642×10^{-13}	1.74536×10^{-13}
	STMM time (s)	37	24
	Total time (s)	80	66
Windows 7 64 bit	$\|\boldsymbol{S}^\dagger\boldsymbol{S}-\boldsymbol{I}\|_{max}$	3.16025×10^{-13}	1.89627×10^{-13}
	STMM time (s)	184	29
	Total time (s)	256	83
Ubuntu 32 bit	$\|\boldsymbol{S}^\dagger\boldsymbol{S}-\boldsymbol{I}\|_{max}$	8.48794×10^{-14}	1.57441×10^{-13}
	STMM time (s)	636	140
	Total time (s)	939	342
Windows 7 32 bit	$\|\boldsymbol{S}^\dagger\boldsymbol{S}-\boldsymbol{I}\|_{max}$	3.19174×10^{-13}	1.99567×10^{-13}
	STMM time (s)	670	176
	Total time (s)	994	447

Output files in the directory `./Data/` reach 81 MB. Environment to use SMatrAn is regulated well, if it is completed to here. A flow of input/output files for the respective executable files will be explained after the next chapter. S-matrix for other optical structure can be also calculated by understanding how to set input parameters.

[Go to table of contents.] [Go to home.]

4 Development schedule and history

[Go to table of contents.] [Go to home.]

5 LICENSE AGREEMENT

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Whenever numerical results obtained from SMatrAn are published in scientific/engineering journals or proceedings, cite the following paper [1]: T. Usuki, M. Saito, M. Takatsu, R.A. Kiehl, and N. Yokoyama, "Numerical analysis of ballistic-electron transport in magnetic fields by using a quantum point contact and a quantum wire," Phys. Rev. B **52**, pp.8244–8255 (1995). If possible, also cite one of the proceedings [2]: T. Usuki, "Frequency domain analysis for optical propagation using Yee lattice," Proceedings of 2013 URSI International Symposium on Electromagnetic Theory (EMTS), pp.503–505 (2013).

Bibliography

[1] T. Usuki *et al.*, Phys. Rev. B **52**, 8244 (1995).

[2] T. Usuki, Proceedings of 2013 URSI International Symposium on Electromagnetic Theory (EMTS), 503–505 (2013).

[3] T. Usuki, *unpublished.*

[4] David Yuk Kei Ko and J. C. T. Inkson, Phys. Rev. B **38**, 9945 (1988).

[5] Patrick A. Lee and Daniel S. Fisher, Phys. Rev. Lett. **47**, 882 (1981).

[6] V. T. Renard *et al.*,Phys. Rev. Lett. **100**, 186801 (2008).

[7] Kjetil M. D. Hals *et al.*,Phys. Rev. Lett. **105**, 207204 (2010).

[8] Bobo Liu, R. Akis and D. K. Ferry, J. Comput. Electron. **13**, 950 (2014).

[9] Yoshitaka Watanabe,"Fast", "Large" and "Accurate" are justice (*in Japanese*).

[10] For another example, the finite element method (FEM) attains a high-accuracy shape for the solution of a partial differential equation.

[11] An authority on numerical analysis also commented progress in C99, p.19 of Ichizo Ninomiya *et al.*, ISBN 4-320-12088-4 (2014, *in Japanese*).

[12] For an example, 'Mersenne Twister' was implemented in <random>.