GPU libraries speed performance analysis for RCWA simulation matrix operations

Jingxiao Xu[a], Martin D. B. Charlton[a]

[a]University of Southampton, University Rd, Southampton SO17 1BJ, United Kingdom

ABSTRACT

Rigorous Coupled Wave Analysis (RCWA) method is highly efficient for the simulation of diffraction efficiency and field distribution patterns in periodic structures and textured optoelectronic devices. GPU has been increasingly used in complex scientific problems such as climate simulation and the latest Covid-19 spread model. In this paper, we break down the RCWA simulation problem to key computational steps (eigensystem solution, matrix inversion/multiplication) and investigate speed performance provided by optimized linear algebra GPU libraries in comparison to multithreaded Intel MKL CPU library running on IRIDIS 5 supercomputer (1 NVIDIA v100 GPU and 40 Intel Xeon Gold 6138 cores CPU). Our work shows that GPU outperforms CPU significantly for all required steps. Eigensystem solution becomes 60% faster, Matrix inversion improves with size achieving 8x faster for large matrixes. Most significantly, matrix multiplication becomes 40x faster for small and 5x faster for large matrix sizes.

Keywords: GPU computing, RCWA, Matrix operation, eigensystem, MKL library

1. INTRODUCTION

Rigorous Coupled-Wave Analysis (RCWA) is an electromagnetic solver to obtain reflection and transmission of a passive periodic dielectric structure. This method is widely used to analyze components incorporating photonic crystal (PhC) structure or diffractive optical elements such as PhC VCSEL, Patterned Sapphire Substrate-LED and solar cells [1][2][3]. RCWA is a Fourier domain method which means the EM fields are solved in Frequency domain space instead of real space. Dielectric function, Electric and Magnetic fields must be Fourier transformed, and represented as harmonic series. In order to make computational solution practical the number of harmonics must be truncated meaning the representation of the structure is approximate, and solution accuracy reduced. Larger and slower matrix calculations will be involved if the periodic structure has complex geometry or many layers. In practice, researchers require a full spectral system response as function of angle of incidence / emission. This is achieved by running hundreds of separate simulations sweeping wavelengths and angle of the incident wave [4]. The RCWA simulation process may last for hours and days depending on the resolution and range of sweep. The aim of this work is to determine methods and bottlenecks which can be used to speed up realistic device simulations with required level of accuracy for applications such as Photonic Crystal VCSEL and Photonic Crystal micro-LED applications.

RCWA algorithm involves numerous matrix operations such as matrix multiplication/inversion, 2D matrix Fast Fourier Transform, eigensystem etc. Parallel computing is an obvious way of improving speed efficiency of matrix operations. Many scientific studies have benefited from applying General-Purpose Graphics Processing Units (GPGPU) to their simulation. Examples include financial applications[5], deep learning **Error! Reference source not found.** and many other parallelizable programs. In this paper a GPU accelerated FDTD solver was developed and optimized providing 15X overall speedup over conventional CPU [6].

Furthermore we investigate the speed performance of commonly available GPU and CPU libraries in resolving each required matrix operation (Eigensystem solution, Matrix inversion / multiplication etc.)

The Compute Unified Device Architecture (CUDA) programming model is a platform developed by NVIDIA to allow programmers to easily utilize GPU processing and is optimized for NVIDIA GPU architectures. CUDA provides many linear algebra computing libraries covering most of the required matrix operations to implement the RCWA algorithm. However Eigensystem solution is not easily implemented and is often slow and inefficient in CUDA. Consequently other GPU accelerated libraries have emerged to plug this gap. In particular MAGMA is effectively the GPU implementation of

Physics and Simulation of Optoelectronic Devices XXXI, edited by Bernd Witzigmann, Marek Osiński, Yasuhiko Arakawa, Proc. of SPIE Vol. 12415, 124150O
 · 0277-786X · doi: 10.1117/12.2650112

the well established LAPACK maths library. MAGMA is designed for heterogeneous architecture and its routines may utilize both CPU and GPU resources [8], allowing blended CPU / GPU computing.

2. RCWA ALGORITHM

RCWA algorithm is to solve Maxwell's equations in Fourier space. It only solves periodic structure as Fourier transform assumes the variables to be periodic which makes it perfect to analyze photonic crystal designs. The unit cell in figure 1 is constructed in RCWA. Firstly, the problem region is divided into three layers which are reflection region, device region (may contain many layers) and transmission region. In each layer, the light propagation properties are calculated and then connected via scattering matrix method. Finally, the reflected and transmitted field are obtained given the incident wave Fourier domain profile.

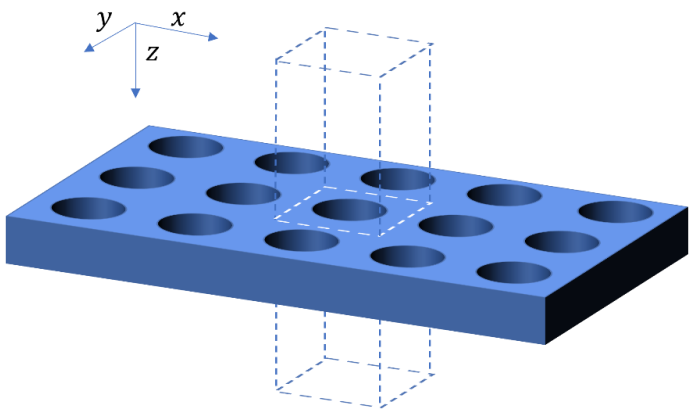

Figure 1. Schematic of an air hole photonic crystal layer.

2.1 RCWA layer solution

RCWA algorithm start from Maxwell's equation.

$$\nabla \times E = k_0 \mu_r \tilde{H} \tag{1}$$

$$\nabla \times \tilde{H} = k_0 \varepsilon_r E \tag{2}$$

The magnetic field $\tilde{H} = j\eta_0 \tilde{H}$. Then fields and properties of the material (permittivity and permeability) are transformed into Fourier space.

$$E_i(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} S_i e^{-j(k_x x + k_y y)} \tag{3}$$

$$\tilde{H}_i(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} U_i e^{-j(k_x x + k_y y)} \tag{4}$$

$$\epsilon_r(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} a_{m,n} e^{j(mT_x x + nT_y y)} \tag{5}$$

$$\mu_r(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} b_{m,n} e^{j(mT_x x + nT_y y)} \tag{6}$$

Where $k_i = k_i^{inc} - mT_i$. T_i is reciprocal space (Fourier space) lattice unit length. $T_i = \frac{2\pi}{\Lambda_i}$. Λ_i is periodicity in i direction. i represents direction x or y. m is any integer. The curl equations in Maxwell's equations are expanded.

$$\frac{\partial \widetilde{H_z}}{\partial y} - \frac{\partial \widetilde{H_y}}{\partial z} = k_0 \epsilon_r E_x \tag{7}$$

$$\frac{\partial \widetilde{H_x}}{\partial z} - \frac{\partial \widetilde{H_z}}{\partial x} = k_0 \epsilon_r E_y \tag{8}$$

$$\frac{\partial \widetilde{H_y}}{\partial x} - \frac{\partial \widetilde{H_x}}{\partial y} = k_0 \epsilon_r E_z \tag{9}$$

$$\frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} = k_0 \mu_r \widetilde{H}_x \tag{10}$$

$$\frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x} = k_0 \mu_r \widetilde{H}_y \tag{11}$$

$$\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} = k_0 \mu_r \widetilde{H}_z \tag{12}$$

Two first differential equations are obtained by substitute (3)-(6) into (7)-(12) and eliminate z filed components.

$$\frac{\mathrm{d}}{\mathrm{d}\tilde{z}} \begin{bmatrix} \boldsymbol{u_x} \\ \boldsymbol{u_y} \end{bmatrix} = \boldsymbol{Q} \begin{bmatrix} \boldsymbol{s_x} \\ \boldsymbol{s_y} \end{bmatrix} \tag{13}$$

$$\frac{\mathrm{d}}{\mathrm{d}\tilde{z}} \begin{bmatrix} \boldsymbol{s_x} \\ \boldsymbol{s_y} \end{bmatrix} = \boldsymbol{P} \begin{bmatrix} \boldsymbol{u_x} \\ \boldsymbol{u_y} \end{bmatrix} \tag{14}$$

Then a second differential is deduced.

$$\frac{\mathrm{d}^2}{\mathrm{d}\tilde{z}^2} \begin{bmatrix} \boldsymbol{s_x} \\ \boldsymbol{s_y} \end{bmatrix} = \boldsymbol{\Omega^2} \begin{bmatrix} \boldsymbol{s_x} \\ \boldsymbol{s_y} \end{bmatrix} \tag{15}$$

Where $\Omega^2 = \mathrm{PQ}$.

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{K}_x [[\mu_r]]^{-1} \boldsymbol{K}_y & [[\varepsilon_r]] - \boldsymbol{K}_x [[\mu_r]]^{-1} \boldsymbol{K}_x \\ \boldsymbol{K}_y [[\mu_r]]^{-1} \boldsymbol{K}_y - [[\varepsilon_r]] & -\boldsymbol{K}_y [[\mu_r]]^{-1} \boldsymbol{K}_x \end{bmatrix} \tag{16}$$

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{K}_x [[\varepsilon_r]]^{-1} \boldsymbol{K}_y & [[\mu_r]] - \boldsymbol{K}_x [[\varepsilon_r]]^{-1} \boldsymbol{K}_x \\ \boldsymbol{K}_y [[\varepsilon_r]]^{-1} \boldsymbol{K}_y - [[\mu_r]] & -\boldsymbol{K}_y [[\varepsilon_r]]^{-1} \boldsymbol{K}_x \end{bmatrix} \tag{17}$$

Where $\boldsymbol{K_x}$ and $\boldsymbol{K_y}$ are diagonal matrices with elements are normalized wave vectors $k_x(m,n)/k_0$ and $k_y(m,n)/k_0$. k_0 is free space wave vector. $k_0 = \frac{2\pi}{\lambda_0}$. $[[\mu_r]]$ and $[[\varepsilon_r]]$ are 2D convolution matrices of permeability and permittivity in Fourier space. After calculating the eigensystem solution. The general solution of equation (7) is written as

$$\begin{bmatrix} \boldsymbol{s_x} \\ \boldsymbol{s_y} \\ \boldsymbol{u_x} \\ \boldsymbol{u_y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{W} & \boldsymbol{W} \\ -\boldsymbol{V} & \boldsymbol{V} \end{bmatrix} \begin{bmatrix} e^{-\boldsymbol{\lambda}\tilde{z}} & 0 \\ 0 & e^{\boldsymbol{\lambda}\tilde{z}} \end{bmatrix} \begin{bmatrix} \boldsymbol{c}^+ \\ \boldsymbol{c}^- \end{bmatrix} \tag{18}$$

Where $\boldsymbol{W}$ and $\boldsymbol{\lambda^2}$ are eigenvectors and eigenvalues of matrix Ω^2. V is obtained by $\boldsymbol{V} = \boldsymbol{QW\lambda^{-1}}$. $\boldsymbol{c}^+ = \boldsymbol{W^{-1}s}^+$ and $\boldsymbol{c}^- = \boldsymbol{W^{-1}s}^-$. $\boldsymbol{s}^+$ and $\boldsymbol{s}^-$ are Fourier coefficients of initial electric field amplitudes in forward and backward direction.

2.2 Scattering matrix

To connect solutions of multi layers, scattering matrix method is used for high numerical stability [9]. Scatter matrix is generally used to connect input and output ports of a system. Each layer i has its own scattering matrix $\boldsymbol{S_i} = \begin{bmatrix} \boldsymbol{S_{11}^i} & \boldsymbol{S_{12}^i} \\ \boldsymbol{S_{21}^i} & \boldsymbol{S_{22}^i} \end{bmatrix}$.

To distinguish from the electric field Fourier coefficient $\boldsymbol{s}$, matrix and submatrix of scattering matrix uses capital $\boldsymbol{S}$.

$$\mathrm{S}_{11}^{\mathrm{i}} = \mathrm{S}_{22}^{\mathrm{i}} = (\mathrm{A_i\text{-}X_iB_iA_i^{-1}X_iB_i})^{-1}(\mathrm{X_iB_iA_i^{-1}X_iA_i\text{-}B_i}) \tag{19}$$

$$\mathrm{S}_{12}^{\mathrm{i}} = \mathrm{S}_{21}^{\mathrm{i}} = (\mathrm{A_i\text{-}X_iB_iA_i^{-1}X_iB_i})^{-1}\mathrm{X_i}(\mathrm{A_i\text{-}B_iA_i^{-1}B_i}) \tag{20}$$

Where $\boldsymbol{A_i} = \boldsymbol{W_i^{-1}W_0} + \boldsymbol{V_i^{-1}V_0}, \boldsymbol{B_i} = \boldsymbol{W_i^{-1}W_0} - \boldsymbol{V_i^{-1}V_0}$ and $\boldsymbol{X_i} = e^{-\lambda_i k_0 L_i}$

A global S matrix [S] is used to connect incident, reflected and transmitted waves.

$$\begin{bmatrix} \mathrm{c}_{\mathrm{ref}}^{-} \\ \mathrm{c}_{\mathrm{trn}}^{+} \end{bmatrix} = [\mathrm{S}] \begin{bmatrix} \mathrm{c}_{\mathrm{inc}}^{+} \\ 0 \end{bmatrix} \tag{21}$$

$$[\boldsymbol{S}] = [\boldsymbol{S_{ref}}] \otimes [\boldsymbol{S_1}] \otimes [\cdots] \otimes [\boldsymbol{S_i}] \otimes [\boldsymbol{S_{trn}}] \tag{22}$$

Then the electric field can be reconstructed using Eqn. (3) with an incident field and connecting multilayer S-matrix with Redheffer Star Product [9]. The Redheffer Star product is defined as

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \otimes \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \tag{23}$$

$$\boldsymbol{C_{11}} = \boldsymbol{A_{11}} + \boldsymbol{A_{12}}[\boldsymbol{I} - \boldsymbol{B_{11}A_{22}}]^{-1}\boldsymbol{B_{11}A_{21}} \tag{24}$$

$$\boldsymbol{C_{12}} = \boldsymbol{A_{12}}[\boldsymbol{I} - \boldsymbol{B_{11}A_{22}}]^{-1}\boldsymbol{B_{12}} \tag{25}$$

$$\boldsymbol{C_{21}} = \boldsymbol{B_{21}}[\boldsymbol{I} - \boldsymbol{A_{22}B_{11}}]^{-1}\boldsymbol{A_{21}} \tag{26}$$

$$\boldsymbol{C_{22}} = \boldsymbol{B_{22}} + \boldsymbol{B_{212}}[\boldsymbol{I} - \boldsymbol{A_{22}B_{11}}]^{-1}\boldsymbol{A_{22}B_{12}} \tag{27}$$

The incident field profile is known. Reflected $\boldsymbol{s_{ref}}$ and transmitted $\boldsymbol{s_{trn}}$ waves are easily obtained.

$$\boldsymbol{c_{ref}^{-}} = \boldsymbol{S_{11}c_{inc}^{+}} \tag{28}$$

$$\boldsymbol{c_{trn}^{+}} = \boldsymbol{S_{21}c_{inc}^{+}} \tag{29}$$

$$\boldsymbol{s_{ref}} = \boldsymbol{W_{ref}c_{ref}^{-}} \tag{30}$$

$$\boldsymbol{s_{trn}} = \boldsymbol{W_{trn}c_{trn}^{+}} \tag{31}$$

3. METHOD

3.1 Matrix operations

A basic RCWA algorithm was implemented in C++ to investigate time cost of each type of matrix operation. Eigensystem solution is used in obtaining equation (18). Matrix inversion is calculated in equation (16), (17) and most equations in scattering matrix method. And matrix multiplication is used all over the algorithm in obtaining single layer solution and computing scattering matrix. Geometry of the example is the single layer air hole model illustrated in fig 1. The concrete material has relative permittivity $\varepsilon_r = 9.0$. While other area is air. Periodicities in x and y direction are both $0.6\mu m$. Diameter of the air hole is $0.3\mu m$. For generality, the data type used in the RCWA program is double complex as permittivity of materials may have complex value. The CPU version employs industry standard Intel MKL library for matrix operation. To start off, we benchmark performance of the MKL library for key computational steps (as identified above) required in the RCWA implementation. Results (figure 2) shows that eigensystem solution, matrix multiplication and inversion are the most time expensive operations. The eigensystem contributes over 50 percent of total time at high harmonics. Each of matrix multiplication and inversion takes around 20%. All together the three operations account for over 90 percent of the total runtime.

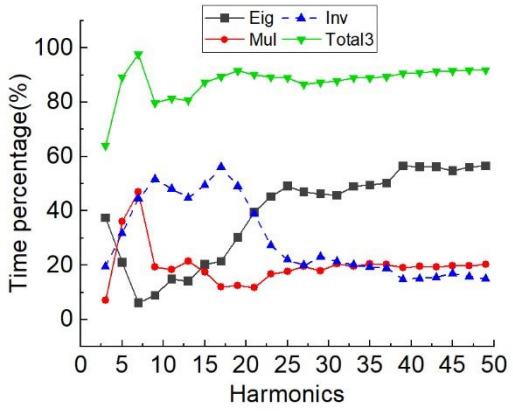

Figure 2. Time cost of eigensystem solution, matrix multiplication and inversion. And the sum of these operations against total runtime represented in percentage.

3.2 GPU library

To reduce total RCWA simulation time, we start with the three most time-consuming operations (eigensystem, matrix multiplication and inversion operations) and compare available GPU library performance against Intel MKL CPU library. . Experiments are conducted on Iridis 5 supercomputer with 40-cores CPU and 1 NVIDIA Tesla V100 GPU. Details of routines used and methods are as follows:

Eigensystem: For a matrix A, where there is a number λ and vector **x** such that $\boldsymbol{Ax} = \lambda \boldsymbol{x}$, λ is called the eigenvalue of A, and the corresponding vector $\boldsymbol{x}$ the eigenvector. In MKL and MAGMA LAPACK, the eigensystem is solved using ZGEEV routine for a general matrix[10]**Error! Reference source not found.**. For simple real symmetric matrix and Hermitian matrix, accelerated algorithm DSYEV and ZHEEV can be used. However, CUDA libraries does not support general (dense non symmetric) matrix eigenvalue calculation as is required for RCWA algorithm[11].

Matrix multiplication: ZGEMM is the routine used to solve general complex matrix multiplication. MKL and CUBLAS both have routine ZGEMM3M, an optimized routine for complex matrix multiplication which provides 25% speedup compared with ZGEMM [13]. While MAGMA only have ZGEMM routine.

Matrix inversion: In MKL and MAGMA LAPACK, a matrix is firstly LU factorised with ZGETRF routine. Then it is inversed using ZGETRI. However, CUSOLVER library does not implement ZGETRI routine. We must use ZGETRS which solve linear equation $A*X=B$. Let B equals to identity matrix. The X is solved to be the inverse of A.

Table 1. The library and routine used for the experiment.

	Eigensystem	Multiplication	Inversion
CPU	MKL(ZGEEV)	MKL(ZGEMM3M)	MKL(ZGETRF+ZGETRI)
GPU	MAGMA(ZGEEV)	MAGMA(ZGEMM) CUBLAS(ZGEMM3M)	MAGMA(ZGETRF+ZGETRI) CUSOLVER(ZGETRF+ZGETRS)

The number of Fourier coefficients, (also called the harmonics) h, is very important and has huge impact on speed of RCWA solution. It defines how many frequency components are used to represent the Fourier transformed dielectric function and EM fields. Consequently it greatly affects the accuracy of solution results. $h = 2N + 1$ where N is the maximum value of m or n in equation (3)-(6). The size of matrix to be stored in computer memory is exponentially related to the number of harmonics. Consequently Eigenvalue solution and matrix multiplication / inversion operations require solution of matrices of size h^2 x h^2. Therefore for RCWA simulation with realistic number of harmonics (ranging from 7-70), matrix operations have size ranging from 100*100 to 10k*10k.

To analyses performance benefits, we test solution speed for the key matrix operation with matrix size ranging from 100*100 to 10k*10k, (step size 100 between 100*100 to 1k*1k and 1000 between 1k*1k to 10k*10k). Matrices values are initially randomized, but once generated identical matrixes are used to compare libraries.

4. PERFORMANCE

4.1 Eigensystem

In RCWA algorithm, the eigen system requires full solution of both eigenvalues and eigenvectors. The matrix is dense and non-symmetric (the worst type!) and can only be solved on GPU with MAGMA GEEV routines.

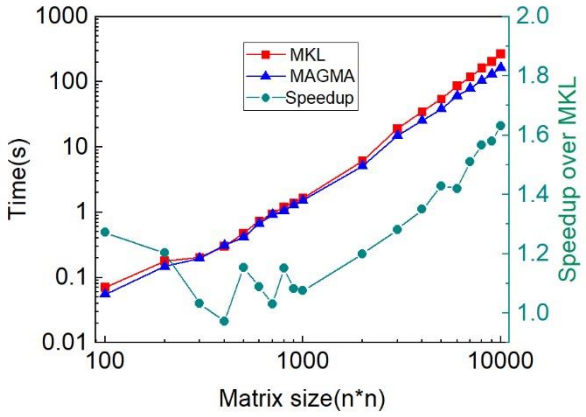

Figure 3. Runtime of eigensystem solver for matrix ranging from 100*100 to 10k*10k. The GPU routine speedup is also shown on the right-y axis.

MAGMA GEEV routine is only partially accelerated and utilizes both CPU and GPU. The speed difference between MAGAM and MKL is not significant for matrix size smaller than 1k, however as matrix size increases, MAGMA speedup over MKL shows an increasing trend. For 10k*10k matrix, MAGMA achieved over 60% speed increase. Although modest, this speed increase is beneficial for RCWA where Eigensystem solution must be performed thousands of times over.

4.2 Matrix multiplication

Matrix multiplication is commonly used in the RCWA algorithm especially in calculating scattering matrix in each layer and performing Redheffer Star Product. Similar to eigensystem solution, general complex matrix is always used in multiplication routines.

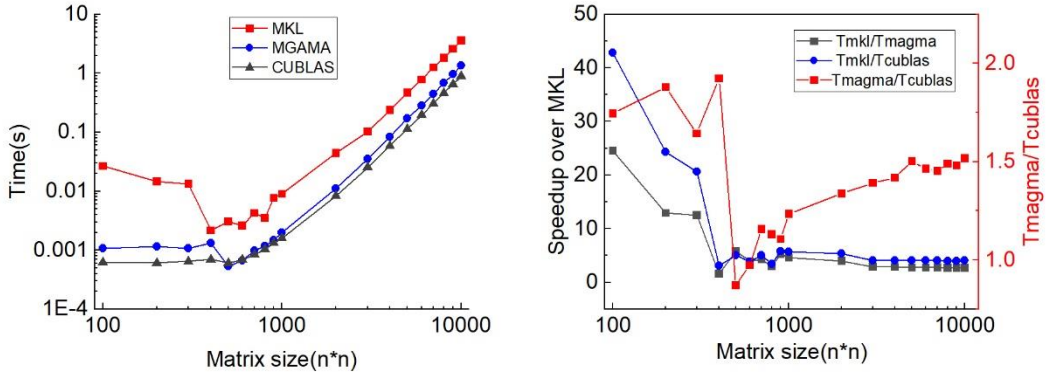

Figure 4. Runtime of matrix multiplication for matrix ranging from 100*100 to 10k*10k. The right-hand side is the speedup of CUBLAS and MAGMA over MKL.

Both MAGMA and CUBLAS outperforms MKL significantly. For small size matrices, all CPU and GPU libraries performs steadily due to runtime being so quick that overhead and non-computational operations dominate the whole process. When matrix size is greater than 500*500, CUBLAS and MAGMA libraries shows 3-5X speedup over MKL, and the trend is expected to remain steady. Meanwhile, CUBLAS is found to be 1.5X faster than MAGMA for large matrix. Interestingly GPU speedup does not keep increasing with matrix size. This may be due to MKL library implementing

efficient parallelization / distribution of matrix multiplication operation across the available 40-CPU cores on our IRIDIS system.

4.3 Matrix inversion

Matrix inversion operation is less frequently used than multiplication. However it has more single operation time than matrix multiplication. Processing a 10k*10k matrix multiplication requires 0.89s to 3.6s. While inverting a 10k*10k matrix takes at least 2.5s with MAGMA and 13.2s using MKL library.

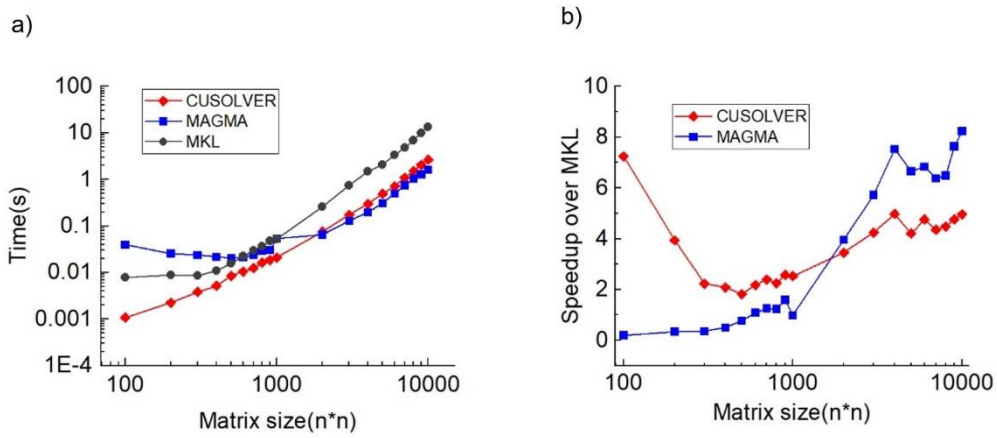

Figure 5. a) Total run time comparison between GPU and CPU RCWA programs. b) Speedup comparison between CSOLVER and MAGMA over MKL.

MAGMA inversion routine achieves a maximum 8X speedup over MKL for 10k*10k matrix. And the speedup is projected to increase rapidly with matrix size. CUSOLVER speedup over MKL also shows an increasing trend but with a slower rate. For matrix larger than 2k*2k, MAGMA consumes less time to process inversion than CUSOLVER.

5. CONCLUSION

In this paper, the computationally expensive matrix operations in RCWA algorithm are implemented and compared between CPU and GPU using available high-performance CPU (MKL) and GPU (MAGMA, CUSOLVER, CUBLAS) linear algebra / maths libraries. Eigensystem solution including eigenvalues and eigenvectors, matrix multiplication and inversions utilize over 90% CPU time in RCWA simulation. GPU implementation of matrix inversion and multiplication provide huge speed increase benefits around 4-8x against CPU MKL library. Eigensystem solution achieved 60% increase in speed utilizing GPU calculation. Overall using GPU libraries is an efficient way to speedup matrix operations and can be used to reduce RCWA simulation time and could be extended to speedup other simulation algorithms involving large numbers of matrix operations.

REFERENCES

[1] Joannopoulos, J. D.; Johnson, S. G.; Winn, J. N. & Meade, R. D. (2008), Photonic Crystals: Molding the Flow of Light (Second Edition), Princeton University Press.

[2] Torrijos-Morán, L., Griol, A. & García-Rupérez, J. Slow light bimodal interferometry in one-dimensional photonic crystal waveguides. Light Sci Appl 10, 16 (2021). https://doi.org/10.1038/s41377-020-00460-y

[3] Sharee J. McNab, Nikolaj Moll, and Yurii A. Vlasov, "Ultra-low loss photonic integrated circuit with membrane-type photonic crystal waveguides," Opt. Express 11, 2927-2939 (2003)

[4] Shi, J., Pollard, M.E., Angeles, C.A. et al. Photonic crystal and quasi-crystals providing simultaneous light coupling and beam splitting within a low refractive-index slab waveguide. Sci Rep 7, 1812 (2017). https://doi.org/10.1038/s41598-017-01842-w

[5] Scott Grauer-Gray, William Killian, Robert Searles, and John Cavazos. 2013. Accelerating financial applications on the GPU. In Proceedings of the 6th Workshop on General Purpose Processor Using Graphics Processing Units (GPGPU-6). Association for Computing Machinery, New York, NY, USA, 127–136. https://doi.org/10.1145/2458523.2458536

[6] T. Gale, M. Zaharia, C. Young and E. Elsen, "Sparse GPU Kernels for Deep Learning," SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, Atlanta, GA, USA, 2020, pp. 1-14, doi: 10.1109/SC41405.2020.00021.

[7] Francés, Jorge & Bleda, Sergio & López, M. & Martínez Guardiola, Francisco & Márquez, Andrés & Neipp, Cristian & Beléndez, Augusto. (2012). Analysis of periodic anisotropic media by means of split-field FDTD method and GPU computing. Proceedings of SPIE - The International Society for Optical Engineering. 10.1117/12.929545.

[8] Stanimire Tomov, Jack Dongarra, Marc Baboulin, Towards dense linear algebra for hybrid GPU accelerated manycore systems, Parallel Computing, Volume 36, Issues 5–6,2010, Pages 232-240, ISSN 0167-8191, https://doi.org/10.1016/j.parco.2009.12.005.(https://www.sciencedirect.com/science/article/pii/S0167819109001276)

[9] Raymond C. Rumpf, "Improved Formulation of Scattering Matrices for Semi-Analytical Methods That Is Consistent with Convention," Progress In Electromagnetics Research B, Vol. 35, 241-261, 2011. doi:10.2528/PIERB11083107, http://www.jpier.org/PIERB/pier.php?paper=11083107

[10] Developer reference for Intel® oneapi math kernel library - C (no date) Intel. Available at: https://www.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top.html (Accessed: January 24, 2023).

[11] Nath R, Tomov S, Dongarra J. An Improved Magma Gemm For Fermi Graphics Processing Units. The International Journal of High Performance Computing Applications. 2010;24(4):511-515. doi:10.1177/1094342010385729

[12] Using the CUSOLVER API (2022) cuSOLVER API Reference. Available at: https://docs.nvidia.com/cuda/cusolver/ (Accessed: January 24, 2023).

[13] Using the cuBLAS API (2022) cuBLAS. Available at: https://docs.nvidia.com/cuda/cublas/ (Accessed: January 24, 2023).