Technical Report

Fast and Massively Distributed Electromagnetic Solver for Advanced HPC Studies of 3D Photonic Nanostructures

Mayeul d'Avezac[1] *and Nicolae C. Panoiu*[2]

[1]*Research Software Development, Research IT Services*
University College London, Malet Place, London WC1E 6BT, United Kingdom

[2]*Department of Electronic and Electrical Engineering,*
University College London, Torrington Place, London WC1E 7JE, United Kingdom

Abstract

Advanced research in optical materials has shown that rapid breakthroughs in device applications go hand-in-hand with the development of efficient, computationally powerful software tools. Thus, high fabrication costs of complex photonic nano-structures and devices make it imperative to have access to computational tools based on efficient algorithms, which can greatly reduce the design-fabrication-testing cycle. In this context, computational methods based on the multiple-scattering formalism have become a primary means to model a variety of scientific and engineering problems. As part of this project, we have transformed OPTIMET – a software based on the multiple-scattering matrix method for solving the problem of electromagnetic waves scattering from arbitrary distributions of particles – from a HPC code that runs efficiently on a University level HPC system to a state-of-the-art HPC software that can be deployed and used on national level HPC facilities such as ARCHER. In particular, the code has been developed along the following directions: *i*) implemented an auto-distribution mechanism for all the computational work required to construct and solve the system of linear equations for the expansion coefficients of the electromagnetic field, the main physical quantities computed with OPTIMET and *ii*) dramatically advanced the functionality of OPTIMET by implementing an efficient, parallelized iterative-scheme complemented by a state-of-the-art acceleration engine. A key outcome of this work was that by adding iterative solvers in conjunction with employing the rotation-coaxial translation decomposition for the matrix–vector multiplication involving the system matrix, the complexity of the problem has been greatly reduced, from $O(N^3 n_{max}^6)$ to $O(N^2 n_{max}^3 N_{iter})$, where N is the number of particles, n_{max} is the truncation order for the vector spherical wave functions, and N_{iter} is the number of iterations needed to reach convergence.

1. Science background

In order to facilitate the understanding of the main capabilities of OPTIMET (OPTIcal METa-materials), we briefly present in this section the numerical method on which the code is based. Thus, the main steps of the multiple-scattering matrix (MSM) method can be described as follows: First, the incident, scattered, and internal (inside the scatterers) fields are expanded in Fourier series. The basis functions in 2D and 3D are multipole functions and vector spherical wave functions (VSWF), respectively [1,2,3,4]. Then, one constructs a system of linear equations whose unknowns are the Fourier coefficients of the scattered field. These coefficients are the main unknowns of the scattering problem as they can be used to compute most of the main physical quantities characterising the wave interaction, namely, the electromagnetic field distribution both inside and outside the scatterers, and the scattering, absorption and the extinction cross-sections.

This brief description of the MSM method shows that the main part of this algorithm consists in constructing and solving a system of linear equations whose unknowns are the Fourier coefficients of the scattered field. The matrix defining this system, also called the *transfer matrix* (or T-matrix), is completely defined by the location, shape and material parameters of the particles. The T-matrix of the system has a block structure, the blocks consisting of single-particle T-matrices and matrices that describe inter-particle interactions (coupling between particles). As a consequence of this block structure of the T-matrix, the MSM method can be easily applied to clusters with different number and distribution of particles. Equally important, the obvious scalability with the number of particles renders the MSM algorithm particularly suitable for parallelisation.

Mathematical Formulation of the Multiple Scattering Matrix Method

Consider a cluster of N particles being illuminated by a monochromatic plane electromagnetic wave. The origin of the co-ordinate system of the cluster is O and to each particle (j) one associates a co-ordinate system with the origin, O_j. The only constraint imposed on the location of the particles is that the circumscribing spheres of the particles do not overlap, *i.e.* $|\boldsymbol{R}_{jl}| = |O_j - O_l| \geq \rho_j + \rho_l, j, l = 1,2, \ldots, N, j \neq l$, where, as per ***Figure 1***, ρ_j is the radius of the smallest sphere containing the jth particle.

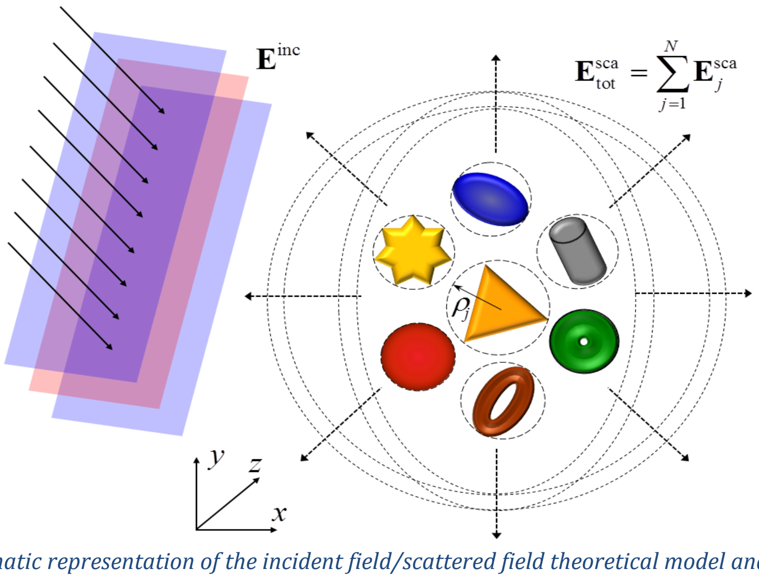
$\mathbf{E}^{\text{inc}}$
$\mathbf{E}_{\text{tot}}^{\text{sca}} = \sum_{j=1}^{N} \mathbf{E}_j^{\text{sca}}$
ρ_j
y
z
x

Figure 1. Schematic representation of the incident field/scattered field theoretical model and 3D structure of a cluster of arbitrary scatterers.

The solution of the source-free Maxwell equations in 3D can be expanded in a complete basis of VSWFs, $\left(\boldsymbol{M}_{mn}^{(1)}, \boldsymbol{N}_{mn}^{(1)}\right)$ and $\left(\boldsymbol{M}_{mn}^{(3)}, \boldsymbol{N}_{mn}^{(3)}\right)$. In a practical implementation, the series is truncated to a certain order, n_{max}. Let $\sum_{mn} \overset{\text{def}}{=} \sum_{n=1}^{n_{max}} \sum_{m=-n}^{n}$; then, the incident, scattered and internal electric fields can be expressed as:

<u>incident field</u>

$$\boldsymbol{E}_0^{inc}(\boldsymbol{R}) = \sum_{mn}\left[a_{mn}\boldsymbol{M}_{mn}^{(1)}(k\boldsymbol{R}) + b_{mn}\boldsymbol{N}_{mn}^{(1)}(k\boldsymbol{R})\right], \tag{1}$$

internal field

$$\boldsymbol{E}^{int}(\boldsymbol{R}_j) = \sum_{mn}\left[c_{mn}^{j}\mathbf{Rg}\boldsymbol{M}_{mn}^{(1)}(k_j\boldsymbol{R}_j) + d_{mn}^{j}\mathbf{Rg}\boldsymbol{N}_{mn}^{(1)}(k_j\boldsymbol{R}_j)\right], \quad R_j < \rho_j, \quad (2)$$

scattered field

$$\boldsymbol{E}^{sca}(\boldsymbol{R}_j) = \sum_{mn}\left[p_{mn}^{j}\boldsymbol{M}_{mn}^{(3)}(k\boldsymbol{R}_j) + q_{mn}^{j}\boldsymbol{N}_{mn}^{(3)}(k\boldsymbol{R}_j)\right], \quad R_j > \rho_j, \quad (3)$$

where k and k_j are the wave vectors in the embedding medium and inside the jth particle, respectively, $\boldsymbol{R}$ and $\boldsymbol{R}_j$, $j = 1,2,\dots,N$, are the position vectors of a point, P, in the co-ordinate systems O and O_j, respectively. The magnetic field is subsequently obtained from the electric field as $\boldsymbol{H} = \boldsymbol{\nabla} \times \boldsymbol{E}$. The unknown expansion coefficients of the internal and scattered fields, (c_{mn}^{j}, d_{mn}^{j}) and (p_{mn}^{j}, q_{mn}^{j}), respectively, can be expressed in terms of the *known* expansion coefficients of the incident monochromatic plane wave, (a_{mn}^{j}, b_{mn}^{j}), by imposing at the surface of the particles the boundary conditions satisfied by the electric and magnetic fields.

a. Single Particle Analysis

Let $\mathbf{a}^j = \left[a_1^j, a_2^j, \dots, a_{u_{max}}^j\right]^{\mathrm{T}}$, with $u_{max} = n_{max}^2 + 2n_{max}$, be a column vector, and define similarly $\mathbf{b}^j$, $\mathbf{c}^j$, $\mathbf{d}^j$, $\mathbf{p}^j$ and $\mathbf{q}^j$. Here, $(\mathbf{a}^j, \mathbf{b}^j)$, $(\mathbf{c}^j, \mathbf{d}^j)$ and $(\mathbf{p}^j, \mathbf{q}^j)$ represent the Fourier expansion coefficients of the incident, internal and scattered fields, respectively, defined with respect to O, relative to O_j, $j = 1,2,\dots,N$. Moreover, if one assumes that the system contains only the jth particle, a compact matrix representation of the Fourier coefficients of the electromagnetic fields, both inside and outside the particle, can be cast as:

$$\begin{bmatrix}\mathbf{p}^j\\ \mathbf{q}^j\end{bmatrix} = \mathbf{T}^j\begin{bmatrix}\mathbf{a}^j\\ \mathbf{b}^j\end{bmatrix} = \begin{bmatrix}\mathbf{T}^{11} & \mathbf{0}\\ \mathbf{0} & \mathbf{T}^{22}\end{bmatrix}^j\begin{bmatrix}\mathbf{a}^j\\ \mathbf{b}^j\end{bmatrix}, \qquad \begin{bmatrix}\mathbf{c}^j\\ \mathbf{d}^j\end{bmatrix} = \mathbf{Q}^j\begin{bmatrix}\mathbf{a}^j\\ \mathbf{b}^j\end{bmatrix} = \begin{bmatrix}\mathbf{Q}^{11} & \mathbf{0}\\ \mathbf{0} & \mathbf{Q}^{22}\end{bmatrix}^j\begin{bmatrix}\mathbf{a}^j\\ \mathbf{b}^j\end{bmatrix}. \quad (6)$$

Here, $\mathbf{T}^j$ and $\mathbf{Q}^j$, $j = 1,2,\dots,N$, are the *scatterer-centred* T-matrix of the jth particle and a matrix that relates the Fourier coefficients of the inside and outside fields, respectively. Note that the block-diagonal structure of the $\mathbf{T}^j$ and $\mathbf{Q}^j$ matrices is valid only for spherical particles.

b. Multiple Particles System

The central idea of the Foldy–Lax multiple scattering theory [5] is that in the vicinity of any scatterer j in the N particle system, there is a local field that is the superposition of the incident field and the field scattered by all the other scatterers in the system, except the scatterer itself. Using the definition of the single-particle T-matrix, $\mathbf{T}^j$, $j = 1,2,\dots,N$, the Fourier expansion coefficients of the scattered and incident fields, $(\mathbf{p}^j, \mathbf{q}^j)$ and $(\mathbf{a}^j, \mathbf{b}^j)$, respectively, obey the following equation:

$$\begin{bmatrix}\mathbf{p}^j\\ \mathbf{q}^j\end{bmatrix} = \mathbf{T}^j\left\{\begin{bmatrix}\mathbf{a}^j\\ \mathbf{b}^j\end{bmatrix} + \sum_{l\neq j}\begin{bmatrix}\mathbf{p}^{lj}\\ \mathbf{q}^{lj}\end{bmatrix}\right\}, \qquad j = 1,2,\dots,N. \quad (7)$$

The coefficients $(\mathbf{a}^j, \mathbf{b}^j)$ and $(\mathbf{p}^{lj}, \mathbf{q}^{lj})$, where $\mathbf{p}^{lj} = \left[p_1^{jl}, p_2^{lj}, \dots, p_{u_{max}}^{lj}\right]^{\mathrm{T}}$, and similarly for $\mathbf{q}^{lj}$, need to be expressed in terms of the known expansion coefficients of the incoming field and $(\mathbf{p}^j, \mathbf{q}^j)$, respectively, which are calculated in the system with the origin in O. This is achieved by using the vector translation-addition theorem [1,3]. Thus, let us consider the matrix coefficients,

$$\alpha^{(j,l)} = \begin{bmatrix} A^{(3)}(k\boldsymbol{R}_{jl}) & B^{(3)}(k\boldsymbol{R}_{jl}) \\ B^{(3)}(k\boldsymbol{R}_{jl}) & A^{(3)}(k\boldsymbol{R}_{jl}) \end{bmatrix}, \qquad j,l = 1,2,\dots,N,\ \ j \neq l, \tag{8a}$$

$$\beta^{(j,0)} = \begin{bmatrix} A^{(1)}(k\boldsymbol{R}_{j0}) & B^{(1)}(k\boldsymbol{R}_{j0}) \\ B^{(1)}(k\boldsymbol{R}_{j0}) & A^{(1)}(k\boldsymbol{R}_{j0}) \end{bmatrix}, \qquad j = 1,2,\dots,N, \tag{8b}$$

where $A^{(z)}(k\boldsymbol{R})$ and $B^{(z)}(k\boldsymbol{R})$, $z = 1,3$, represent the vector translation-addition coefficients, which are evaluated using an efficient recursive implementation as proposed in [3]. Let $\mathbf{f}^j = \left[\mathbf{p}^j\ \mathbf{q}^j\right]^{\mathrm{T}}$ and $\mathbf{A} = [\mathbf{a}\ \mathbf{b}]^{\mathrm{T}}$. Then, the single-particle scattering expansion coefficients, $\mathbf{f}^j$, can now be expressed in terms of $\mathbf{A}$ and $\mathbf{f}^l$, $j,l = 1,2,\dots,N, j \neq l$, as:

$$\mathbf{f}^j = \mathbf{T}^j \beta^{(j,0)}\,\mathbf{A} + \mathbf{T}^j \sum_{l \neq j} \alpha^{(j,l)} \mathbf{f}^l. \tag{9}$$

In terms of the known Fourier expansion coefficients of the incident field, $\mathbf{A}$, the resulting system of linear equations for the N particle system for the unknown Fourier expansion coefficients of the scattered field, $\mathbf{f}^j, j = 1,2,\dots,N,$ can be expressed in the following form:

$$\begin{bmatrix} \mathbf{I} & -\mathbf{T}^1\alpha^{(1,2)} & -\mathbf{T}^1\alpha^{(1,3)} & \cdots & -\mathbf{T}^1\alpha^{(1,N)} \\ -\mathbf{T}^2\alpha^{(2,1)} & \mathbf{I} & -\mathbf{T}^2\alpha^{(2,3)} & \cdots & -\mathbf{T}^2\alpha^{(2,N)} \\ -\mathbf{T}^3\alpha^{(3,1)} & -\mathbf{T}^3\alpha^{(3,2)} & \mathbf{I} & \cdots & -\mathbf{T}^3\alpha^{(3,N)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\mathbf{T}^N\alpha^{(N,1)} & -\mathbf{T}^N\alpha^{(N,2)} & -\mathbf{T}^N\alpha^{(N,3)} & \cdots & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}^1 \\ \mathbf{f}^2 \\ \mathbf{f}^3 \\ \vdots \\ \mathbf{f}^N \end{bmatrix} = \begin{bmatrix} \mathbf{T}^1\beta^{(1,0)}\mathbf{A} \\ \mathbf{T}^2\beta^{(2,0)}\mathbf{A} \\ \mathbf{T}^3\beta^{(3,0)}\mathbf{A} \\ \vdots \\ \mathbf{T}^N\beta^{(N,0)}\mathbf{A} \end{bmatrix}. \tag{10}$$

The system (10) can be expressed in a compact matrix form, $\mathbf{SF} = \mathbf{U}$, where $\mathbf{F} = [\mathbf{f}^1, \mathbf{f}^2, \dots, \mathbf{f}^N]^{\mathrm{T}}$, $\mathbf{U} = \left[\mathbf{T}^1\beta^{(1,0)}\mathbf{A}, \mathbf{T}^2\beta^{(2,0)}\mathbf{A}, \dots, \mathbf{T}^N\beta^{(N,0)}\mathbf{A}\right]^{\mathrm{T}}$ and $\mathbf{S}$ denotes the scattering matrix of the system. *Constructing the system matrix* $\mathbf{S}$ *and solving Eq. (10) represent the core parts of the MSM method.*

2. Features developed during the eCSE

2.1. Parallelization

As explained in the preceding section, OPTIMET is nothing more than a code to invert a (very specific) matrix. As such, to complete work-package 1, we first turned to Scalapack, a standard Fortran/C library with parallel linear algebra capabilities, including solving for arbitrary linear systems of complex variables. With the refactoring work done (see ***Sec. 4.3***), adding Scalapack as a solver and Matrix-Vector multiplication operation became a fairly straight-forward exercise in (i) adding a dependency to the build system when compiling OPTIMET with MPI and (ii) adding few C++ wrappers to simplify Scalapack' and MPI's C routines. The performance improvements are given in a later section.

The main difficulty arose when, for larger and more complex systems, creating the matrix became a prohibitive bottleneck (this actually became true only when the iterative solvers from Trilinos were integrated into OPTIMET). To parallelize this procedure, we rely on Scalapack's ability to create and transform block-cyclic matrices with blocks of different sizes. We also rely on the condition that all the scatterers are described by the same spherical harmonics basis. In this case, the matrix can be decomposed into a set of equal-sized blocks, each describing the interaction between two particles. By creating a distributed block-cyclic matrix where the block size is a multiple of the size of an interaction-matrix, we can easily ensure that each process creates only parts of the final matrix. Then we request Scalapack to transform the block cyclic matrix to one where the blocks are sized for performance rather than convenience.

2.2. Iterative Solvers

Because of the underlying assumptions, mainly with regards to approximating scatterers with homogeneous spherical particles, the solution to the linear system solved by OPTIMET converges relatively fast, especially in the case of dielectric materials. Hence iterative solvers could well prove much more efficient than direct solvers. We chose to use the family of solvers which already exist in Trilinos, more specifically in the Belos subpackage. This gives us the ability to explore a variety of state-of-the-art iterative solvers, rather than spend time implementing our own. One of the interesting features of the Belos solvers is that they only require from users a vector in a Trilinos object and a function to perform matrix-vector multiplications. The format of the underlying matrix does not matter, indeed it does need exist explicitly. As a result, we could integrate the solvers from Trilinos directly with the Scalapack distributed matrices implemented above. The majority of the work in this package came down to implementing glue code between Trilinos, Scalapack and OPTIMET, as well as some refactoring to allow for separated solvers in OPTIMET. The performance gains are discussed below; in practice, they proved to be quite large.

2.3. Rotation-coaxial translation decomposition of the matrix-vector operation

The main operation during an OPTIMET simulation is simply a matrix-vector multiplication, where the vector describes the fields impinging upon each scatterer, and the matrix the scattered field from each scatterer projected on the basis centered at each scatterer. Hence an efficient simulation requires an efficient matrix-vector multiplication. Prior to this eCSE, OPTIMET relied on an explicit dense matrix of complex numbers, without any specific structure, which could help solve the linear system.

However, the physical operation the matrix represents can be simplified into a set of multiplication by sparse matrices, resulting in fewer floating point operations for a given number of scatterers and associated local basis sets [6,7]. The original operation represents the following physics: (i) scatter the impinging wave onto a scatterer A using a basis centered at A; (ii) project the scattered field from the basis centered at A onto a basis centered at another scatterer B. The impinging wave at B is the sum of the input incident wave and the waves scattered by all other particles. Step (i) is a diagonal operation when described in a basis set centered at the scatterer. A naive implementation of (ii), however, results in a dense matrix-vector operation. It can be shown that it could be decomposed in the following set of sparse operations [6,7]: (ii-a) rotate the local basis onto a set of Cartesian coordinates such that the z-axis is in the direction A to B, (ii-b) project the rotated basis onto a basis local to B (with the same orientation of the co-ordinate system but at a different origin), (ii-c) rotate back onto the original Cartesian directions. (ii-a) is the inverse operation of (ii-c), whereas (ii-b) is a translation along the line connecting A and B. These three operations are sparse when expressed between spherical harmonic basis. Even though we have now expressed the matrix-vector multiplication using four explicitly separate operations, rather than one, it still requires fewer actual floating point operations by a factor equivalent to the number of elements in the truncated spherical harmonic basis sets [6]. To be more specific, as a result of this matrix decomposition, the computational effort is reduced from $O(N^3 n_{max}^6)$ to $O(N^2 n_{max}^3 N_{iter})$, where N is the number of particles, n_{max} is the truncation order for the vector spherical wave functions, and N_{iter} is the number of iterations needed to reach convergence.

The coefficients for the sparse rotation and translation matrices can be obtained using recurrence relations [7]. These are expedient and need only be computed once per simulation. However, we have found that they are numerically unstable. More specifically, for a given precision, the order of the operations in the recurrence relation will influence the actual result. The numerical instability increases for higher-order spherical harmonics. In practice, it becomes noticeable only for components which bear little spectral weight. For instance, we have found instances for spherical harmonics of degree 20 where the rotation coefficients differ by 0.6×10^{-7} when switching from double to long-double precision. The same problem exists with the translation coefficients. We also expect that the translation-addition coefficients of the original implementations also suffer from the same issues. Further work should likely study this problem in more detail, e.g. by implementing the recurrence relations in a language with floating points of arbitrary precision. Since these coefficients are computed once and for all at the beginning of a simulation, we chose to implement the relationships internally using "long double" (80 or 128 bits, rather than 64 bits, depending on infrastructure and compilers) and truncate the result back to 64 bits. This approach offers a good compromise between the complexity of the implementation, accuracy and performance.

2.4. Parallel rotation-coaxial translation decomposition operation

The matrices involved in OPTIMET have no particular structure, which could speed up computations. However, the algorithmic requirements for OPTIMET do offer some grounds for efficient parallelization: (i) the main operation limiting performance is a simple matrix-vector multiplication; (ii) the iterative solvers from Belos can be parameterized to use any custom function implementing a vector operation of some kind; (iii) the matrix does not change throughout the algorithm; and (iv) only vector quantities are of physical interest.

The parallelization scheme is illustrated in ***Figure 2***. The vector quantities are composed of the vector potentials projected onto local basis sets $(\o_j^{(i)}, \psi_j^{(i)})$ centered at each particle i. For simplicity, we distribute the vectors by assigning a contiguous range of particles to each process. For instance, the elements $(\o_j^{(1)}, \psi_j^{(1)})$ and $(\o_j^{(2)}, \psi_j^{(2)})$ are assigned to process 1, $(\o_j^{(3)}, \psi_j^{(3)})$ and $(\o_j^{(4)}, \psi_j^{(4)})$ to process 2 and so on. The matrix-vector operation is then decomposed into: an operation A which requires strictly local input data but may lead to distributed output data, and an operation B which may require data from other processes but will lead to output data needed only locally. For instance, the interaction of particle 2 with 3 can be computed from the data owned by process 2 but will lead to data required by process 1. It can be done in operation A by process 2. Similarly, given the input data from particle 3, process 1 can compute the output it owns resulting from the interaction of particles 1 and 3. In practice, we use a banded matrix, where the diagonal elements correspond to A (local input data), and the off-diagonal components to B (local output data). However, the algorithm is implemented for any binary partition of the scattering matrix (in blocks particle interactions). The algorithm proceeds as follows:

1. Communicate local data needed for operation B by other processes
2. Compute operation A using local input data
3. Communicate non-local output data from operation A
4. Compute operation B using data communicated in step 1
5. Reduce local output data received from other processes in step 3 and computed locally in step 2

Thus, the two communications patterns are interleaved with computations. This scheme could be applied to any matrix-vector multiplication. However, it is also uniquely well suited to the fast matrix multiplication operation described previously, since the interactions between pairs of particles are essentially independent from one another.

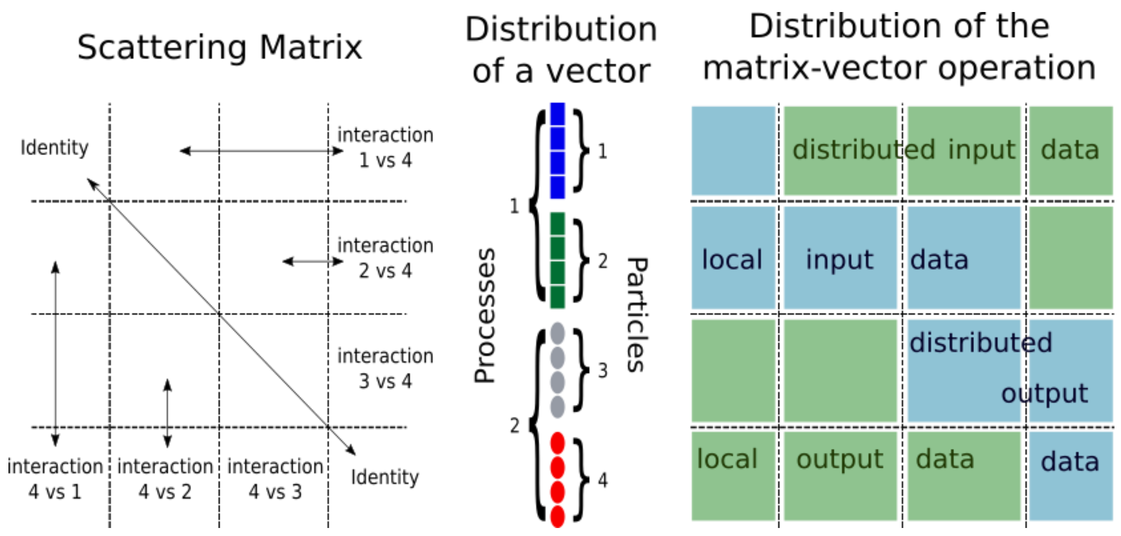

Figure 2*: Parallelization scheme for the rotation-coaxial translation decomposition of the matrix-vector operation. Each process owns all the basis coefficients for a contiguous range of particles. The matrix-vector operation is composed of blocks representing the interaction of particles (i, j). Furthermore, it can be decomposed over those operations that require coefficients owned locally, but may output non-local coefficients, and those that may require non-local coefficients, but always output contributions to local coefficients.*

3. Technical work undertaken during the eCSE

OPTIMET is a C++ code originally developed in Prof. Panoiu's group by Dr. Ahmed Al-jarro and Dr. Claudiu Biris. At the beginning of the eCSE project, it was a serial code written with somewhat less common and less efficient C++ design pattern, and using a large number of the dependencies. It had no test suite. It used at least three different linear algebra implementations, including an in-house developed set of routines implementing standard matrix-vector multiplication.

As such, our first objective was to streamline both the framework surrounding the code and the code itself. This work took place primarily at the start of the project. However, it went on as further need arose. The following is a non-exhaustive list of the changes:

- OPTIMET was added to GitHub, making full use of its version control feature and issue management system.
- Automated regression tests were created, exercising the code as it existed on the first day of the project. This ensures that changes to the code during refactoring will not affect the output of the code.
- A unittest framework was added, to allow for test-driven development of new features, as well as for the more heavily refactored parts of the code.
- OPTIMET is integrated into UCL's automated testing framework (Jenkins instance).
- The dependencies were consolidated to include Eigen for linear algebra, boost and f2c (for a C version of Amos included in the code) for special mathematical functions, HDF5 for output serialization. Currently, GSL is also needed because of a legacy matrix multiplication routine in the code.
- We removed much of the in-house developed linear algebra routines in favor of using Eigen throughout the code. These routines relied on matrices implemented as non-contiguous array of arrays. This sort of idiom is less well suited to high performance computing.
- We introduced namespaces to the code, and removed instances of a less-standard C++ idiom whereby classes of static member functions where used to the same effect.
- We replaced the systematic multi-step resource allocation idiom with RAII (resource allocation is initialization).
- We replaced naked pointers with the smart pointers and naked arrays with STL or Eigen containers, where relevant.
- We instituted a type hierarchy for the major numerical types.
- We replaced the chains of Boolean returns with exceptions when appropriate.
- We re-implemented the set of routines computing the translation-addition coefficients both to simplify the code by centering it on the mathematical recursion it implements and to remove hard-coded values limiting the scope of the recurrence relations and the applicability of the method.
- We added a framework to the build system to automatically download dependencies when they are not found on the system.

That is to say a fair share of the project – about a third of the time – was spent retooling and refactoring the code to meet standard research software development practices. We believe that this initial heavy investment was highly rewarding during the later stages of the projects when we added new features to the code. However, it is in subsequent years and features that we expect the full benefit of the work on code sustainability to be felt.

There is still work needed before OPTIMET can be said to fully meet high research software development standards. The following is an unordered list of technical works that would improve the sustainability of the code in the future:

- Remove fully the in-house developed linear algebra routines (and the GSL dependencies, as a result).

- Find a replacement for Amos (used for special mathematic functions with complex arguments). Amos is based on a Fortran code with typical awkward API and incorrect results in given simple corner cases (e.g. at the origin). This will remove the dependency on F2C.
- Refactor input and output, and the Geometry module.
- Add logging.
- Add timing counters to the logging system, to automate some aspect of profiling the code.
- Profile the serial and MPI code for memory and speed bottlenecks.

4. Performance and benchmarks

OPTIMET is composed of somewhat orthogonal components, namely the construction of the system matrix, the matrix-vector operation and the solvers. The parallel direct solver requires Scalapack, but the iterative solvers can use either the Scalapack matrix-vector operation or the decomposed sparse matrix-vector operation (RCTD). We present the performance of the matrix-vector multiplications first. Then we show that the iterative solvers far outperform the previously used direct solvers in the case of OPTIMET. Finally, we demonstrate the significant, total performance gain from changing the solver, switching to RCTD and code parallelization.

All the benchmarks are run using the Scalapack implementation bundled with the Intel MKL on Dell C6100 dual processors with six cores per processor, and QDR InfiniPath chip-to-chip connectivity. The problems we will investigate consist of equally sized particles in vacuum, with a radius of a quarter of the wavelength, and a complex relative permittivity of (13.1 + i). By choosing a complex value for the dielectric constant we ensure that we test both the metallic and dielectric cases. These values are within the range of practical applications, but do not correspond to a specific material. The particles are arranged on a face-centered lattice. For a given number of particles, we grow a crystal along 001, 010, and 100 directions equally. The objective here is to have a problem that scales well with the number of particles without changing its symmetry and thus try and ensure difficulty of the problem remains the same. In other words, we aim for a set of problems with stable condition numbers. We expect that other geometries and materials will yield a somewhat different convergence behavior. However interesting, an extensive study of performance with respect to material properties and geometries falls outside the remit of this project. The iterative solvers are set to break for a tolerance of 10^{-6}.

4.1. Performance of the matrix-vector operation

Figure 3 compares the performance of RTCD and the Scalapack multiplication on a single process. From a purely performance oriented point of view, RCTD become very advantageous for larger basis sets of spherical harmonics. Most notably, it is advantageous in the range that most applications will require. The increase in the cost of performing a single operation with respect to the number of particles is roughly quadratic, as indicated by the regression. This result is only a first step towards a performance model, and should be understood as such. Since Scalapack is a heavily optimized library, whereas the RCTD operations are currently implemented for correctness rather than speed (and thus not yet comprehensively profiled or optimized), these preliminary results are highly encouraging. Furthermore, the Scalapack implementations we have had access to, including those available on ARCHER, are unable to address matrices larger than 2GB (at least for some operations, including matrix manipulations), most likely because of the type of the underlying indices. This happens for relatively small problem sizes, e.g. for systems of 30 particles with spherical harmonics of degree 5 or less or 10 particles with spherical harmonics of degree 10 or less. OPTIMET and our implementation of RCTD rely on a well-defined type hierarchy. It does not suffer of this problem for currently feasible applications, and would be trivial to update in the unlikely event that 64bit indexing becomes insufficient.

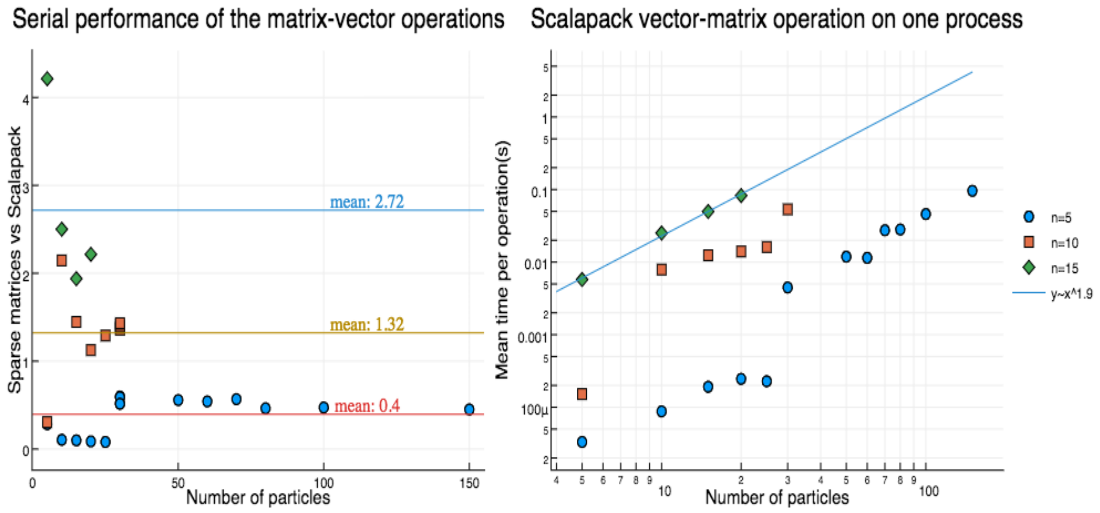

Figure 3: *The left panel shows the speedup between performing a rotation-coaxial translation of the matrix-vector operation (RCTD) versus a Scalapack multiplication on a single process for a set of problems with varying number of particles (on the x-axis) and spherical harmonics cut-off (legend). RTCD becomes more advantageous for basis-sets with larger cut-offs. The right panel shows the amount of time in seconds taken by a single Scalapack matrix-vector operation for the same set of problems.*

We now turn to the parallelization of the RCTD, as reported in ***Figure 4***. It shows that our scheme is efficient up to the limit of distribution (one particle per process) or to the limit of the tests we ran. The ideal parallelization scheme is represented with a solid black line. For larger problem sizes, our scheme achieves very good speedup. However, we would like in future work to probe the limits of the scheme further. We find that the parallelization scheme saturates only at the algorithmic limit, when the distribution of the problem hits the lower limit of one particle per process. We should remark that the largest problem in ***Figure 4*** are mobilising the whole available memory in the case of serial (single node) execution. Finally, the scaling graphs for smaller basis sets (not shown here) are quite similar, though somewhat less efficient. This is likely because the communication and computations do not overlap as well since the latter takes much less time for smaller basis sets. It should be noted that we tested the code performance in a rather challenging situation, namely when the dielectric constant of the particles is complex (metallic particles). In this case one requires a relatively large number of harmonics to reach convergence, as in this case the electromagnetic field is strongly inhomogeneous.

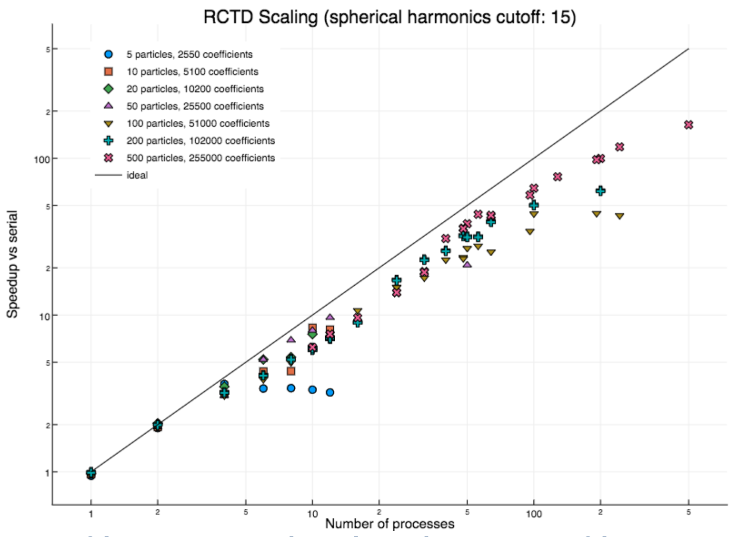

Figure 4*: Parallelization of the rotation-coaxial translation decomposition of the matrix-vector operation with MPI. The ordinate reports the speedup, i.e. the time taken to perform the matrix-vector operation in parallel and per processor versus the time taken in serial. The abscissa reports the number of processes thrown at each problem. Each line represents a problem with a given size (varying number of particles, all spherical harmonics of degree 15 or less). The results show that for larger problems, we have not yet hit parallelization limit beyond which adding more processes would become unhelpful. Note that the current parallelization scheme can distribute the problem only to the lower limit of 1 particle per process. Hence, the 5-particle problem saturates at 5 processes. The number of coefficients corresponds to the size of the vectors in the matrix-vector operations.*

4.2. Performance of the iterative vs. direct solvers

We examine in ***Figure 5*** the performance of an iterative solver (GMRES) provided by Trilinos versus a direct provided by Scalapack. In both cases, we use the matrix-vector operation from Scalapack. For simplicity, we do not optimize for the block-size or grid shape of the block-cyclic matrix, nor for the different flavors of the iterative solvers, nor for the parameters of GMRES. We find that the iterative solver systematically outperforms the direct solver for all problem sizes and number of processes. We also find that the GMRES with RCTD systematically outperforms GMRES with the Scalapack multiplication. This is not surprising since for the basis set consisting of spherical harmonics of order 15 or less, we saw previously that RCTD outperforms Scalapack.

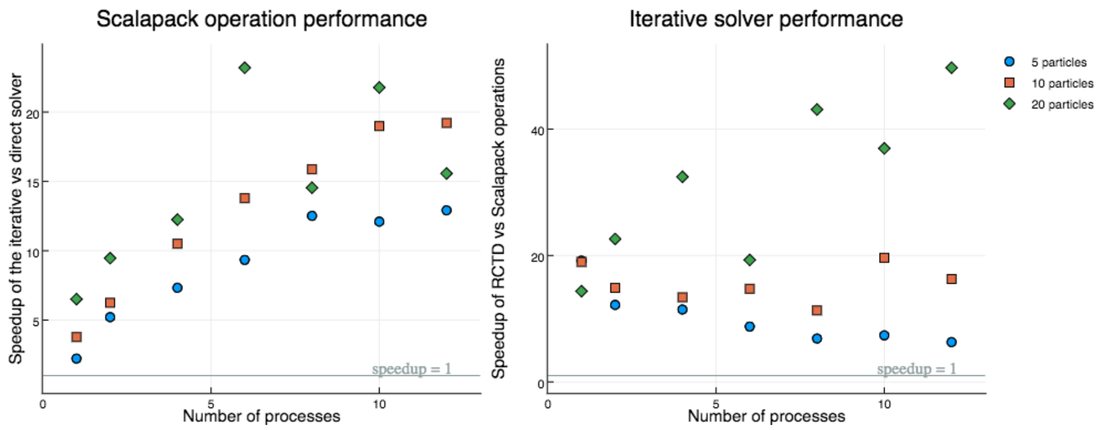

Figure 5*: Performance of the GMRES iterative solver vs. the direct solver from Scalapack. The left panel shows the speedup obtained when going from the direct to the iterative solver with respect to the number of processors, using the Scalapack matrix vector operation. The right panel compares the two matrix operations when using the iterative solver (the direct solver is not available with RCTD). In both cases, we show substantial speedup, especially for larger problem sizes. The basis set is composed of all spherical harmonics of order 15 or less.*

4.3. Performance of OPTIMET when operated in parallel

Finally, we turn to the performance of OPTIMET as a whole, using all the features added during this project, namely RCTD and the GMRES iterative solver, over MPI. The results are presented in ***Figure 6***. We find that for basis sets of spherical harmonics of order 15 or less, the parallelization scales well up to the maximum number processes we were able to throw at our benchmark, that is, until the distribution reaches one particle per process. For 500 particles distributed over 500 cores, we obtain a speedup of 152 over the performance in serial (with the same solver and matrix-vector operation). In practice, we have not yet reached the limitation of the parallelization technique. Since we are comparing to the serial implementation, the results presented in ***Figure 6*** are limited by the amount of memory that a single node can hold. We have also examined the behaviour of larger problems requiring either fatter single nodes, or distribution over several nodes. We find that the solution to a 1000 particle problem will converge within a few seconds on 12 nodes and 144 cores (though it did not complete in 12 hours on 2 nodes, see below). With this achievement, we can claim that the current eCSE has not only made it possible to run much larger problems within reasonable time-scales, but also access problems that were previously limited by memory and by the time needed to create the matrix.

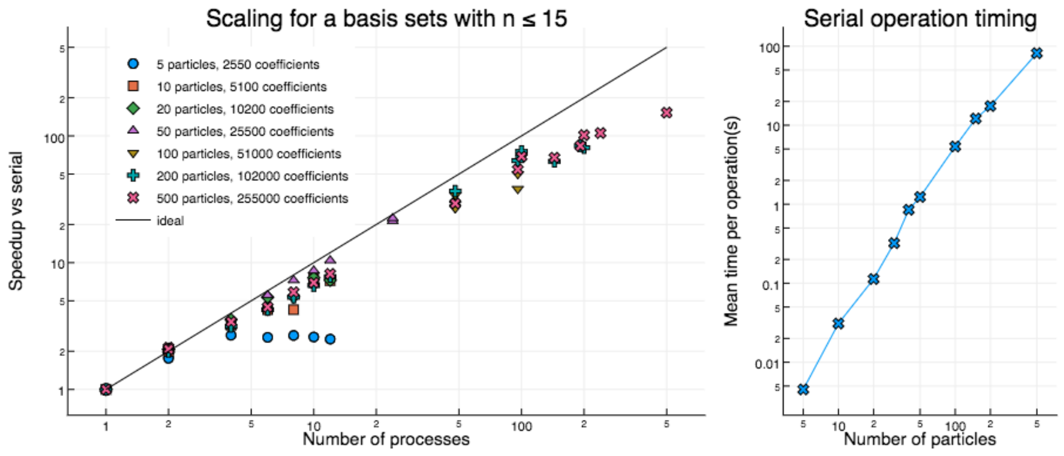

***Figure 6**: Performance of OPTIMET when the iterative solver GMRES from Trilinos in conjunction with the Decomposed Sparse Matrix-Vector Operation. The left panel shows the speedup achieved with respect to the serial code. The right panel shows the time taken by the serial code with respect to the number of particles.*

Finally, it turns out for larger systems that setting up the matrix-vector operators is now more computationally intensive than solving the linear equations. Indeed, the same run of 1000 particles mentioned earlier takes longer on 2 nodes than the 12 hours of wall-time we allowed for it. We expect this is a result of filling the available memory and the time needed for constructing the matrix-vector operation. Setting up the operation for a 1000 particles on 144 cores takes a few minutes, whereas solving the linear system takes a few seconds. It should be noted that setting up the problem grows as the square of the number of particles. Future work will require benchmarking and profiling the setup stage of the matrix-vector operation, rather than just its application. Preliminary profiling results seem to indicate that the current implementation spends much time allocating and deallocating small arrays. This is not unexpected if we consider how we coded the computation of different coefficients for simplicity and accuracy rather than speed. However, it is easy to re-engineer the memorization part of the recurrence coding to limit the number of memory allocations.

5. Conclusion

In this report we have described the structure, code development activities (parallel and serial), characterization tests and a few application examples of OPTIMET. The code has been written and tested for validity and scalability, as illustrated in this report, with strong scalability being proven. In particular, the code has been developed along the following directions: *i*) implemented an auto-distribution mechanism for all the computational work required to construct and solve the system of linear equations for the expansion coefficients of the electromagnetic field, the main physical quantities computed with OPTIMET, and *ii*) dramatically advanced the code functionality by implementing an efficient, parallelized iterative-scheme complemented by a state-of-the-art acceleration engine. A key outcome of this work is that by adding iterative solvers in conjunction with using the rotation-coaxial translation decomposition for the matrix–vector multiplications, the problem complexity has been greatly reduced, from $O(N^3 n_{max}^6)$ to $O(N^2 n_{max}^3 N_{iter})$, where N is the number of particles, n_{max} is the truncation order for the vector spherical functions, and $N_{iter}$is the number of iterations needed to reach convergence. This remarkable improvement of the code efficiency translated to a significant increase of the complexity of the problems that can be tackled by OPTIMET. In particular, now OPTIMET can be used to model a broad array of science and engineering problems, including photonic and plasmonic crystals, absorption engineering, nanoscience and nanotechnology, plasmonics, transformation optics, optical tomography and colloidal chemistry. In particular, two journal papers based on OPTIMET have already been produced by Prof Panoiu's group and his collaborators [8,9], one already published and one submitted.

OPTIMET has been developed for ARCHER, but has also been tested on *Legion*, UCL's main HPC cluster. It is freely available *via* GitHub to the computational electromagnetics community and is installed on ARCHER as well, so that we expect a significant increase in user numbers. The fully verified and tested code will also be made available through a dedicated webpage at UCL and through the program library of *Computer Physics Communications* journal *via* a peer-reviewed journal article. The eCSE support was for 12 months FTE divided between two people, Dr. Mayeul d'Avezac and Dr. Gary Macindoe.

Acknowledgments

This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk).

Bibliography

1. M. Mishchenko, L. Travis, and A. Laci, *Scattering, Absorption and Emission of Light by Small Particles*, (Cambridge University Press 2002).
2. M. I. Mishchenko, G. Videen, V. A. Babenko, N. G. Khlebtsov, and T. Wriedt, "*Comprehensive T-matrix reference database: A 2004–06 update*," J. Quant. Spectr. Rad. Trans. **106**, 304–324 (2007).

3. B. Stout, J.-C. Auger, and J. Lafait, "*A transfer matrix approach to local field calculations in multiple scattering problems*," J. Mod. Opt. **49**, 2129–2152 (2002).
4. C. G. Biris and N. C. Panoiu, "*Second harmonic generation in metamaterials based on homogenous centrosymmetric nanowires*," Phys. Rev. B, **81**, 195102 (2010).
5. M. Lax, "*Multiple scattering of waves*," Rev. Mod. Phys. **23**, 287–310 (1951).
6. N. A. Gumerov and R. A. Duraiswami, "*A scalar potential formulation and translation theory for the time-harmonic Maxwell equations*," J. Comput. Phys. **225,** 206–236 (2007).
7. N. A. Gumerov and R. A. Duraiswami, "*Recursions for the Computation of Multipole Translation and Rotation Coefficients for the 3-D Helmholtz Equation*," SIAM J. Sci. Comput. **25,** 1344–1381 (2004).
8. A. Al-Jarro, C. G. Biris, and N. - C. Panoiu, "*Resonant mixing of optical orbital and spin angular momentum by using chiral silicon nanosphere clusters*," Opt. Express **24**, 6945 (2016).
9. Xiaoyan Y.Z. Xiong, A. Al-Jarro, L. J. Jiang, N. - C. Panoiu, and Wei E.I. Sha, "*Mixing of Spin and Orbital Angular Momenta via Second-harmonic Generation in Plasmonic and Dielectric Chiral Nanostructures*," Phys. Rev. B (submitted).