Optical diagnostics of a single evaporating droplet using fast parallel computing on graphics processing units

D. JAKUBCZYK, S. MIGACZ, G. DERKACHOV, M. WOŹNIAK*, J. ARCHER, and K. KOLWAS

Institute of Physics, Polish Academy of Sciences, Al. Lotników 32/46, 02–668 Warsaw, Poland

We report on the first application of the graphics processing units (GPUs) accelerated computing technology to improve performance of numerical methods used for the optical characterization of evaporating microdroplets. Single microdroplets of various liquids with different volatility and molecular weight (glycerine, glycols, water, etc.), as well as mixtures of liquids and diverse suspensions evaporate inside the electrodynamic trap under the chosen temperature and composition of atmosphere. The series of scattering patterns recorded from the evaporating microdroplets are processed by fitting complete Mie theory predictions with gradientless lookup table method. We showed that computations on GPUs can be effectively applied to inverse scattering problems. In particular, our technique accelerated calculations of the Mie scattering theory on a single-core processor in a Matlab environment over 800 times and almost 100 times comparing to the corresponding code in C language. Additionally, we overcame problems of the time-consuming data post-processing when some of the parameters (particularly the refractive index) of an investigated liquid are uncertain. Our program allows us to track the parameters characterizing the evaporating droplet nearly simultaneously with the progress of evaporation.

Keywords: optical characterisation, microdroplets, light scattering, inverse scattering problem, graphics processing units, GPU, Mie theory, parallel computing.

1. Introduction

Analysis of light scattered by various particles are needed in numerous research endeavours, such as interstellar dust study [1], combustion systems [2], dusty plasmas [3], evaporating liquids [4], colloidal suspensions [5,6], or different biomedical applications [7,8]. It is also vital for nanophotonics research [9,10].

In order to find the light scattering properties, numerous theories, algorithms and models have been developed [11]. Some of their numerical implementations utilize graphics processing units (GPU) technology [12,13]. Scattering theories and algorithms consider different sizes, morphologies, ambient media and other specific properties of the scattering systems. In choosing a particular solution one must consider its relevance and accuracy, as well as the computational efficiency. The trade-off between these factors has resulted in various numerical techniques developed over the past years to reduce the overall time and memory requirements for computation of full wave electromagnetic scattering models [14].

A particularly often studied phenomenon is the light scattering by a spherical particle. It is analytically described by the Mie theory in terms of infinite series of spherical multipole partial waves [14]. The Mie theory is a rigorous solution to divergent free Maxwell equations which assumes that the particle is a homogeneous sphere, of arbitrary size and the incident light is a plane wave [11]. At the beginning of “computer age”, Mie calculations gained a bad reputation for being unstable and time-consuming owing to the extreme computational labour involved in evaluating the series expansion functions such as the Ricatti-Bessel functions [15]. Due to more efficient and powerful computers these days, computational time of the Mie theory has become more and more acceptable. Recently, many stable and efficient methods have been developed for spheres with Mie size parameters high as 10000 [16–18].

In typical scattering applications one faces an inverse problem which requires summing the scattering results over a large number of radii (as in integrating over size distribution), or over a large number of wavelengths (as in integrating across the electromagnetic spectrum) or over a large number of refractive indices (as in inverting scattering measurements to deduce refractive index). Our numerical inversion model is based on fitting of the complete Mie theory predictions (with the look up table method) to the scattering patterns (i.e., scat-
tering intensities versus scattering angle) obtained in the experiment with evaporating droplets. In this way we find the temporal evolution of the droplet radius versus time $a(t)$ (see e.g. [19–21]). It is worth noticing that inversion of a single scattering pattern is not a challenge from computational point of view. However, it becomes so when dealing with a series of scattering patterns recorded as a movie during evaporation of a droplet. Evaporation process of a single droplet may take from few seconds up to hours, depending on the properties of liquid and ambient conditions (atmosphere composition, temperature, pressure, etc.). Therefore, for long series of scattering patterns, data inversion and analysis take long hours of computational time.

In order to resolve this problem, we implemented the Mie scattering calculations and our inversion algorithm on graphics processing units (GPUs) using parallel computing framework, known as CUDA (Compute Unified Device Architecture), introduced by NVIDIA Corporation [22,23]. This technology, initially developed for rendering detailed real-time images for output display, has been widely used not only for their native applications but also for the extensive scientific calculations. CUDA platform enables dramatic increases in computing performance by harnessing the power of the graphics processing units (GPUs). Recently, GPU parallel computations have shown very high performance in several numerical tasks in electromagnetic scattering [24–28]. For example, Iadarola et al. [25] showed how the $T$-matrix algorithm naturally fits into the GPU computing environment by numerical experiments on computing the scattering from an isolated non-axisymmetric particle and from an agglomerate of arbitrary shaped particles. They achieved more than 20x speedup over sequential execution in case of isolated particle and more than 25x speedup for a particle cluster on GPU as compared to CPU. Li et al. [27] have also shown different electromagnetic integral-equation solvers implemented on GPUs using NVIDIA GeForce GTX 570. It is also worth noticing that the code was 2.5 times faster when tested on GeForce GTX 680.

The paper is organised as follows. After the introduction, section 2 presents the experimental setup for droplet levitation and optical analysis. Section 3 presents numerical algorithms, particularly it briefly explains the Mie scattering theory and our lookup table method used for the experimental data inversion. In sect. 4 we present the code parallelisation on GPU. Section 5 presents the experimental result. Section 6 shows the general conclusions and perspectives of the presented work.

2. Experimental setup

In our experiment we measure the temporal evolution of radius of freely suspended evaporating droplets of various liquids with different volatility and molecular weight (glycerine, glycols, water, etc.), as well as mixtures of liquids and diverse suspensions. We use the experimental setup and measurement procedures presented in Fig. 1.

![Fig. 1. The outline of experimental setup and procedures.](image-url)
Optical diagnostics of a single evaporating droplet using fast parallel computing on graphics processing units

Inside the electrodynamic trap, a combination of alternating (AC) and static (DC) electric field constrains a droplet (or particles) to a small volume of “free space”. The trap is kept in a climatic chamber under the chosen temperature and composition of atmosphere. Droplets are injected by on-demand injector and charged on injection by charge separation in the external field.

Two coaxial, counter propagating lasers are used simultaneously for droplet illumination: green (532 nm; 10 mW) p-polarized and red (654 nm; 10 mW) s-polarized. The entirely defocused image is used for droplet sizing. The recorded series of scattering patterns (movie in .avi file format) are processed by fitting complete Mie theory predictions (with gradientless, lookup table method; see bottom part of Fig. 1). It should be noted here that our droplets are very small (i.e., typically smaller than 50 μm in diameter) and hence, as shown in [29,30], they are almost perfectly spherical.

Additionally, since the typical droplet charge in our experimental setup is usually below 5·10^5 elementary charges [31] we are far from the Rayleigh limit [32], and hence droplet deformation caused by Coulombic forces is not observed. Taking these into account, the Mie scattering theory can be applied for the estimation of the droplet size with high accuracy.

The focused image is used for droplet position stabilization with a PID type loop. Simultaneously, DC signal providing stabilization can be used for droplet (particle) weighting (see top part of Fig. 1). Further details of our experimental setup and procedures can be found in Ref. 19 and in references given therein.

3. Numerical algorithms

3.1. Mie scattering theory

Since its introduction in 1908 by Gustav Mie [14], the Mie scattering theory is the most important and widely used theory for the description of the light scattering and absorption properties of spherical particles [11]. The Mie theory describes in an exact manner the interaction between a monochromatic electromagnetic plane wave and a homogeneous, isotropic, nonmagnetic and spherical particle defined by the diameter and the complex refractive index (both absorbing and non-absorbing particles included). It is a solution of the Maxwell’s wave equation(s) using the method of separation of variables, in the spherical coordinates with appropriate boundary conditions. External medium must be non-absorbing. The Mie theory is used as a reference model for nearly all particle light scattering techniques [33]. According to the Mie theory, in the far field (i.e., when kr >> 1), the relations between the incident \( E_i \) and the scattered \( E_s \) electrical fields can be expressed for the two main polarization components \((p:\parallel l \text{ and } s: \perp \text{ to the scattering plane})\) as

\[
\begin{align*}
\left( \begin{array}{c} E_{s\parallel} \\
E_{s\perp} \\
\end{array} \right) &= \frac{e^{ik(r-z)}}{-ikr} \left( \begin{array}{cc} S_2 & S_3 \\
S_3 & S_1 \\
\end{array} \right) \left( \begin{array}{c} E_{i\parallel} \\
E_{i\perp} \\
\end{array} \right),
\end{align*}
\]

where \( k = 2\pi/\lambda \) is the wavenumber of the incident light in the surrounding medium and \( \lambda \) the incident wavelength. The factors \( e^{-ikr} \) and \( e^{ikr}/-ikr \) pertain to the incident plane harmonic wave propagating along the z axis and to the out-going scattered wave respectively. For a spherical particle \( S_3 = S_4 = 0 \) and the scattering amplitude functions \( S_1, S_2 \), are given by

\[
S_1 = \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)} (a_n \pi_n + b_n \tau_n),
\]

\[
S_2 = \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)} (a_n \tau_n + b_n \pi_n),
\]

where the angle dependent functions \( \pi_n \) and \( \tau_n \) are related to the associated Legendre functions \( P_n^m \) of the first kind of order 1 and degree \( n \).

\[
\pi_n = \frac{P_n^n}{\sin \theta}, \quad \tau_n = \frac{dP_n^n}{\sin \theta}.
\]

The expansion coefficients for the scattered field of the transverse magnetic (TM) mode (no radial magnetic field) \( a_n \) and of the transverse electric (TE) mode (no radial electric field) \( b_n \) are given by Ref. 11

\[
a_n = m \frac{\Psi_m(mx)\Psi'_n(x) - \Psi'_m(mx)\Psi_n(x)}{m \Psi_m(mx)\xi'_n(x) - \xi'_m(mx)\Psi_n(x)},
\]

\[
b_n = \frac{\Psi'_m(mx)\Psi'_n(x) - m \Psi'_n(mx)\Psi'_m(mx)}{\Psi'_m(mx)\xi'_n(x) - m \xi'_n(mx)\Psi'_m(mx)},
\]

where \( m \) is the refractive index and \( x = ka \) is the size parameter of a spherical particle with the diameter \( a \) respectively. The functions \( \Psi_n(\rho) \), \( \xi_n(\rho) \) are the Riccati-Bessel functions which are related to the spherical Bessel functions of the first kind \( j_n(\rho) \) and the spherical Hankel function of the first kind \( h_n^{(1)}(\rho) \) as

\[
\Psi_n(\rho) = \rho j_n(\rho),
\]

\[
\xi_n(\rho) = \rho h_n^{(1)}(\rho).
\]

3.2. Mie Scattering Lookup Table Method (MSLTM)

In our experiments we determine the temporal evolution of the radius of single evaporating liquid droplet suspended in the electrodynamic trap, primarily with the Mie Scattering Lookup Table Method (MSLTM) [19]. Our method is based on the fitting of the complete Mie theory predictions (stored
in the lookup table) to the experimentally obtained scattering patterns. During the droplet evaporation we record movies containing series of consecutive scattering patterns (usually 35 frames per second). For each scattering pattern our algorithm finds a solution of the linear least square problem defined as follows

$$\min_{r_n} \sum_{\theta} (I(\theta) - s(r_n) I_T(\theta, r_n))^2,$$  \hspace{1cm} (6)

where $I(\theta)$ corresponds to experimental scattering pattern (i.e. light intensities measured for discrete scattering angles $\theta$, $I_T(\theta, r_n)$ is a matrix of scattering patterns calculated for a given droplet radius $r_n$.

Due to the fact that amplitudes of the experimentally recorded intensities may significantly differ from the numerically predicted ones, it is necessary to use normalization factor $s(r_n)$. This factor, serving as the regularization parameter, is calculated for each droplet radius in the following way

$$s(r_n) = \frac{\sum_{\theta} I(\theta) I_T(\theta, r_n)}{\sum_{\theta} I_T^2(\theta, r_n)}. \hspace{1cm} (7)$$

Figure 2 shows an example of the scattering pattern obtained in experiment and corresponding numerically generated scattering pattern found with our algorithm. The experimental pattern was obtained from the analysis of a single frame of a movie recorded during the evaporation of a microdroplet of diethylene glycol.

In favourable cases (slowly evaporating liquid, refractive index known a priori) our method provides accuracy of droplet radius measurement around ±10 nm [19] for droplet radius varying from 30 μm down to 0.3 μm. As an associated method for MSLTM we use electrostatic weighting. Droplet radius can be, in combination with MSLTM, also inferred from the DC voltage used to balance the droplet weight and maintain the droplet in the (pseudo-)potential minimum of the trap. However, this method lies out of the scope of this paper. It is discussed in details in Ref. 19.

An example of the result obtained with both MSLTM and electrostatic weighting for the evolution (evaporation) of a droplet of diethylene glycol is shown in Fig. 3. Unprocessed temporal evolution of the droplet radius obtained with MSLTM is shown with solid line, while stabilizing DC voltage evolution (corresponding to droplet mass evolution) is shown in solid black dots.

4. GPU programming

GPU is a specialized electronic circuit, initially developed for graphics purposes, with highly parallel architecture. Particularly, the graphics card GeForce GTX 680 used in our laboratory contains 1536 cores with 1006 MHz clock rate and 2 GB of ultrafast (192.2 GB/sec) memory. NVIDIA CUDA (Compute Unified Device Architecture) [22] is a programming platform which allows writing and executing highly parallel C-like code on a GPU. Due to common GPUs limitations, so far, we have used single-precision floating-point format instead of double-precision. We found that it reduces the accuracy of our calculations below 1%, which is fully satisfactory for our application. In case of need for higher precision the selected runs of calculations can be repeated with much more time-consuming double precision.

A function executed on a GPU (called “kernel”) launches a grid of blocks each consisting of many threads.
within each block can cooperate using for example a shared memory or synchronization. Usually an efficient CUDA code launches many threads (tens of thousands or hundreds of thousands) which are required to fully utilize GPU and shared memory blocks. Our numerical solution has been divided into two main procedures: “Generate Pattern” and “Reference Distance”. Flow charts of both procedures are shown in Fig. 4 and Fig. 5, respectively.

“Generate Pattern” procedure calculates lookup table used for data inversion. Firstly, in a single loop, our program calculates the angle-dependent functions $\frac{n}{c_1}$ and $\frac{n}{c_2}$ [Eq. (3)]. Then, in a large nested loop iterated over particle radius and incident wavelength (currently we use two wavelengths but application of many wavelengths is possible), two consecutive loops calculate the scattering coefficients [Eq. (4)] and finally the scattering amplitudes $S_1, S_2$ [Eq. (2)].

In order to calculate the scattering amplitudes, the results of the previous calculations must be used. Time necessary to perform “Generate Pattern” procedure depends not only on the number of radii but also on their absolute values. It is a consequence of the estimation for the minimal required expansion order of the spherical Bessel functions that we used [11]

$$n_{\text{max}} = x_{\text{max}} + 4x_{\text{max}}^{1/3} + 2,$$

where $x_{\text{max}}$ is the limit value of the size parameter (for maximum radius $r_{\text{max}}$). Detailed discussion concerning derivation of the aforementioned limit and its influence to the calculation accuracy can be found in [11,15,34]. The minimal required expansion order is calculated at the very beginning of our program. As an example, time necessary to calculate lookup table containing, for instance, 2000 radii spanning the range between and 3 μm with 1 nm increment step, is approximately twice shorter than the time necessary for preparation of the analogical table between 6 and 8 μm.

The “Reference Distance” procedure performs numerical inversion using lookup table generated by the previous procedure and finds the radius of a sphere corresponding to the experimentally recorded scattering pattern. In the first nested, loop iterated over time (frame number) and particle radius, it calculates regularization parameters given by Eq. (7). Afterwards, the procedure calculates the distance [in the least square sense, Eq. (6)] between the experimental and the theoretical scattering patterns. The latter are normalized by the regularization parameter calculated beforehand. Finally, single loop finds the minimum of the distances (i.e., least square fitting errors) calculated for the various radii of the sphere. The value of the radius corresponding to the theoretical scattering pattern with minimum distance from the experimental one is accepted as the experimental value of the droplet (particle) radius.

For both “Reference Distance” and “Generate Pattern” procedures the parallelization was straightforward. In a serial C implementation, the most of the time was spent in three nested loops. In the two outer loops no communication or synchronization is required; therefore, we assigned each iteration of these loops to a different block on GPU. The
third inner loop performs the reduction across an array (i.e., sums the values into a common buffer). Because the communication and synchronization is now required, we assigned all the computations to a single block, and used shared memory to perform the reduction within the block. Benchmarks showed that 128 is the optimal number of threads per block for both “Reference Distance” and “Generate Pattern”, therefore each block processes data frames in chunks of 128. Maximal performance is achieved when the number of processed data frames is significantly larger than 128, mostly because in that case threads could efficiently add a lot of values into a local buffer (stored in registers) and do an expensive reduction in shared memory only once at the end of computation.

5. Results and discussion

For the numerical calculations we used C-like code running on the NVIDIA GeForce GTX 680 graphics card. Computational times have been compared with analogical code running in Matlab environment (single-core implementation) and also with C-language code running on quad-core CPU. We tested both single-core and quad-core implementations. All the test calculations were performed on the up-to-date desktop computer with Intel Core i7-3770K processor (3.40 GHz clock rate), 16 GB RAM memory and NVIDIA GeForce GTX 680 graphics card.

Figure 6(a) compares the total computational time between the single-core Matlab, single and quad-core CPU and GPU solutions. In the given case we used the lookup table with scattering patterns corresponding to an evaporating microdroplet of glycerine. The total time of evaporation was nearly 500 s. It can be seen that the total computational time linearly depends on the length of the movie and was successfully reduced using GPU technology. The computational time for a 20000-frames-long sequence, was reduced from 1686.20 seconds for calculations in Matlab to 192.89 and 51.04 s in the C language, for the single and the quad-core processors respectively [Fig. 6(a)]. Finally, only 2.02 s was necessary for the GPU. Figure 6(b) shows the relative speedup of the GPU calculations vs. the total length of movie. For the sufficiently long movies the speedup reached ~830, ~95 and ~25 times comparing to single-core Matlab, single-core C code and quad-core C code on CPU, respectively. Smaller speedup for shorter movies is due to the fact that initial procedures (such as reading data files and calculation of the inversion matrix in “Generate Pattern”) cannot be as effectively parallelized as the others (calculation of the regularization parameters and the least square algorithm). Additionally, time used by “Generate Pattern” does not depend on the length of the movie, but only on the input parameters describing the scattering problem. Since the number of radii and the other scattering parameters are not changing from data frame to data frame, this procedure is executed only once at the beginning of the calculations and, therefore, it contributes more significantly to the total computational time of the shortest movies.

Figure 7 shows the comparison of the computational time and the relative speedup between the various implementations of “Generate Pattern” procedure, for the same microdroplet of glycerine as in Fig. 6, but for different look-up tables. In order to calculate lookup tables for this test, we chose a sliding window containing a constant number of 2001 radii with 1 nm increment step. The central value of the window is shown in x axis. It is worth noticing that for “Generate Pattern” alone, the relative speedup of calculations is significantly smaller than for the total computational time (shown in Fig. 6). It reaches only ~50, ~22 and ~16 times comparing to single-core Matlab, single-core C
and quad-core C code respectively. As mentioned above, smaller speedup of “Generate Pattern” procedure is due to the fact that initial procedures (such as reading data files) and the Mie scattering calculations are not as effectively parallelized as the calculation of the regularization parameter and the least square algorithm. The time necessary for these calculations depends only on the input parameters of the procedure, particularly on the size parameter. The larger the size parameter, the longer the calculations. In case of a moving window of the constant length used here, the size parameter increases, thus the total time increases. It is worth noticing that the increase of the speedup is not linear as previously, since for longer expansion series the number of threads per block varies. We can observe characteristic “oscillations” in the speedup when the number of threads is not optimal.

6. Conclusions

Calculations of light scattering by various particles are needed in numerous research areas. Scattering theories and algorithms consider different sizes, morphologies, ambient media and other specific properties of the scattering systems. Facing time-consuming calculations, we have applied GPU accelerated computing technology to improve performance of our numerical methods used in our optical signal processing.
We showed that GPU programming can be effectively applied to inverse scattering problems. The calculations are a part of the larger setup utilized for the characterization of evaporating microdroplets. The recorded series of scattering patterns are processed by fitting complete Mie theory predictions with gradientless, lookup table method. Computational difficulties encountered during the experimental work were associated not with the algorithm complexity but with the number of iterations needed to obtain the entire temporal evolution of the droplet radius. Time necessary for both the Mie scattering calculations and the data inversion with the lookup table method was significantly reduced. The total speedup for the favourable cases reached ~830, ~95 and ~25 times comparing to single-core Matlab, single-core C and quad-core C code running on CPU, respectively. Additionally, we overcame problems of time-consuming data post-processing when some of the parameters (particularly the refractive index) of an investigated liquid are uncertain. With GPU technology, we can easily iterate over the unknown parameter and find the best fit. Very effective parallelization was possible since the calculations for consecutively recorded radii of an evaporating droplet (data frames) are independent. Data exchange was necessary only within the data frame. The ultimate goal was to write a program which from a human operator point of view displays results nearly instantly. For a moderate droplet size of few micrometers, lookup table of 20-nm-step and up to ~300 data frames (usual display resolution) the goal can be considered as accomplished. Faster GPUs can easily further reduce the constraints.

Further improvements of our methods would be aimed at providing solutions for more complex scattering problems. We would like to use different algorithms, such as modification of Mie theory for spheroids [35], T-Matrix method [36] and Discrete Dipole Approximation [37].

Acknowledgements
This work was supported by the National Science Centre, Poland under grants number 2014/13/D/ST3/01882 and 2014/13/B/ST3/04414.

References
19. D. Jakubczyk, G. Derkachov, M. Kolwas, and K. Kolwas, “Combining weighting and Scatterometry: application to a le-
Optical diagnostics of a single evaporating droplet using fast parallel computing on graphics processing units