Asci 98 (55)

A PARALLEL FAST DISCRETE DIPOLE APPROXIMATION FOR LARGE SCALE SIMULATIONS OF ELASTIC LIGHT SCATTERING

ALFONS HOEKSTRA AND PETER SLOOT

Parallel Scientific Computing and Simulation group, Department of Computer Science, Faculty of Mathematics, Computer Science, Physics, and Astronomy, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, the Netherlands, tel 31205257463, fax 31205257490, email alfons@wins.uva.nl, http://www.wins.uva.nl/wins/research/pscs/

Simulation of Elastic Light Scattering from arbitrary shaped particles in the resonance region (i.e. with a dimension of several wavelengths of the incident light) is a long standing challenge. By employing the combination of a simulation kernel with low computational complexity, implemented on powerful High Performance Computing systems, we are now able to push the limits of simulation of scattering of visible light towards particles with dimensions up to 10 micrometer. This allows for the first time the simulation of realistic and highly relevant light scattering experiments, such as scattering from human red - or white blood cells, or scattering from large soot - or dust particles. We use the Discrete Dipole Approximation to simulate the light scattering process. In this paper we report on a parallel *Fast* Discrete Dipole Approximation, and we will show the performance of the resulting code, running under PVM on a 32-node Parsytec CC. Furthermore, as an example we present results of a simulation of scattering from human white blood sells. We investigate the influence of the position of the inner sphere, modeling the nucleus of a Lymphocyte, on the light scattering signals.

Keywords: Computational Science, modelling and simulation of Elastic Light Scattering, parallel computing

1. Introduction

Simulation of Elastic Light Scattering (ELS) from arbitrary shaped particles with dimensions in the order of a few wavelengths, i.e. in the resonance region, is a challenging problem, with many important applications.[1,2] A method which is very well suited for such simulations is the Discrete Dipole Approximation (DDA).[3] The DDA discretises a particle into small subvolumes which are approximated as dipoles. The electromagnetic field on each dipole, due to an incident field and due to the radiation of all other dipoles, is calculated. Next, the scattered field is calculated.

As with all numerical simulations of wave phenomena, the maximum size of the discretisation is bounded by the wavelength ($\Delta x < \lambda/10$). In a previous paper we showed that this bound results, for particles in the resonance region, in DDA models containing up to 10^7 or even 10^8 dipoles.[4] The calculation of the internal field, i.e. the fields on the dipoles, requires the solution of a dense system of $3N$ equations with $3N$ unknowns, where N is the number of dipoles. This is achieved using iterative, conjugate gradient methods, which have an $O(N^2)$ complexity.[4,5] It is obvious that DDA simulations of particles in the resonance region require very powerful computers.

We have parallelised the DDA and showed that it runs very efficiently on distributed memory computers, provided that the number of dipoles per processor is large enough.[4] The parallel DDA is fully scalable and can benefit from the computational power offered by massively parallel systems. However, this parallel DDA still suffers from the $O(N^2)$ complexity which should be reduced in order to accomplish the required simulations with 10^6 or more dipoles. A way to do that, as was shown by Goodman et al.,[6] is to use fast Fourier techniques in the matrix vector products of the conjugate gradient iterations, resulting in a DDA method with $O(N\log N)$ complexity. We call this the *Fast* DDA (FDDA).

We report on a parallel implementation of the FDDA method, and we investigate the performance of this parallel FDDA on the Parsytec CC.

Our major interest is ELS from human white blood cells.[7,8,9,10] We will present results of simulations of a model of a Lymphocyte, a small Human White Blood Cell. The model consists of a sphere with a spherical inclusion. The inner sphere mimics the nucleus of the cell. The influence of the position of the nucleus on the light scattering is investigated.

2. The Discrete Dipole Approximation

The DDA is extensively described in previous papers and major developments of the method were recently reviewed on several occasions.[3,11] Here we will only mention those aspects of the method that are relevant for this paper. Consider an arbitrary particle illuminated by a monochromatic electromagnetic field $\mathbf{E}^0(\mathbf{r})$ with wavelength λ. Our task is to calculate the scattered electric field $\mathbf{E}^s(\mathbf{r})$ in the full solid angle around the particle.

The Discrete Dipole Approximation (DDA) discretises the particle into N, usually equal sub-volumes. The size of a sub-volume, d, must be small enough to ensure that its response to an electromagnetic field is the response of an ideal induced dipole. The size should be in the range $\lambda/20 < d < \lambda/10$, with λ the wavelength of the incident light (see e.g. the discussions in Ref. 12).

The electric field on each dipole, due to an external field and the fields radiated by all other dipoles, must be calculated. Once the electric field on the dipoles is known, the scattered field is calculated by summing the contributions of all dipoles in the far field region. The electric field on dipole i ($1 \le i \le N$), due to the external field $\mathbf{E}^0(\mathbf{r})$ and the field radiated by all other dipoles, is

$$\mathbf{E}(\mathbf{r}_i) = \mathbf{E}^0(\mathbf{r}_i) + \sum_{j \neq i}^{N} \mathbf{F}(\mathbf{r}_i, \mathbf{r}_j) \cdot \mathbf{E}(\mathbf{r}_j) . \qquad (2.1)$$

The 3×3 matrix $\mathbf{F}(\mathbf{r}_i,\mathbf{r}_j)$ describes the radiation from dipole j on dipole i (see e.g. Ref. 4 for an exact definition). Eq. (2.1) defines a set of $3N$ equations for the $3N$ unknowns $(\mathbf{E}_x(\mathbf{r}_i)$, $(\mathbf{E}_y(\mathbf{r}_i)$, $(\mathbf{E}_z(\mathbf{r}_i))$. After solving the matrix equation, the scattered electric field $\mathbf{E}^s$ is calculated by summing the fields, radiated by the dipoles, at the observation point $\mathbf{r}_{obs}$ (which is usually taken at infinity):

$$\mathbf{E}^S(\mathbf{r}_{obs}) = \sum_{i=1}^{N} \mathbf{F}(\mathbf{r}_{obs}, \mathbf{r}_i) \cdot \mathbf{E}(\mathbf{r}_i) . \qquad (2.2)$$

Calculation of the electric field on the dipoles, Eq. (2.1), is the most expensive computational part of the DDA method. From a numerical point of view, this calculation boils down to solving a very large system of linear equations $\mathbf{Ax} = \mathbf{b}$, with $\mathbf{A}$ a $n \times n$ complex symmetric matrix, $\mathbf{b}$ a known complex vector and $\mathbf{x}$ the unknown complex vector. This system of equations is solved using a Conjugate Gradient method. We apply the so-called CGNR method.[4,8] In previous work we have parallelised the CGNR method for distributed memory computers, in the Single Program Multiple Data paradigm,[4] and used it to develop a parallel DDA simulation. Recently Rahola et al. have shown that other Krylov space methods, especially QMR, result in better convergence.[5] Our parallelisation techniques for the CGNR are applicable without change in DDA calculations using e.g. QMR.

In Fig. 1 we indicate the number of floating point operations needed for 1 iteration of the conjugate gradient method in the DDA (the line *direct DDA*), as a function of the number of dipoles N. In order to simulate ELS from human white blood cells, which have diameters up to 16 µm,[8,13] the number of dipoles needs to be in the range 10^5 to 10^8. In Ref. 4 we described a parallel version of the direct DDA, and based on this work we are able to indicate the range of operations which can be performed in less than 10 minutes, when executed on a typical workstation (a Sun Sparcstation 20 at 50 MHz) or on a parallel computer containing 32 PowerPC-604 processors (a Parsytec CC) (see Fig. 1). The demand that the maximum execution time of one iteration is limited to 10 minutes is not completely arbitrary. Typical large DDA models which we currently employ normally need in the order of 100 iterations (unpublished results). The 10 minutes limit would then result in maximum execution times in the order of 17 hours, or approximately an overnight calculation per particle.

It is obvious from Fig. 1 that the direct DDA is too demanding if we wish to simulate ELS from realistic, micron sized particles. Goodman et al. pointed out that due to the property $\mathbf{F}(\mathbf{r}_i,\mathbf{r}_j) = \mathbf{F}(\mathbf{r}_i\text{-}\mathbf{r}_j)$, the matrix vector products which appear in the conjugate gradient iterations can be reformulated as discrete convolutions of electric fields on the dipoles.[6] These convolutions can be calculated in an $O(N\log N)$ complexity (using fast Fourier transformations), which is an enormous reduction in operations as compared to the direct $(O(N^2))$ calculation. The operation count for one iteration of the conjugate gradient method in this *Fast* DDA (FDDA, i.e. DDA using the fast Fourier transforms) is also indicated in Fig. 1. Under the assumption that FDDA can run at approximately the same speed as DDA, this suggests that Goodman's FDDA, when executed on a 32 node CC (i.e. a *low complexity kernel* executed on a *powerful HPC system*) allows to cover a significant range of numbers of dipoles N needed to model realistic, micron-sized particles. We therefore developed a parallel FDDA, using the previously developed parallel version of the direct DDA.

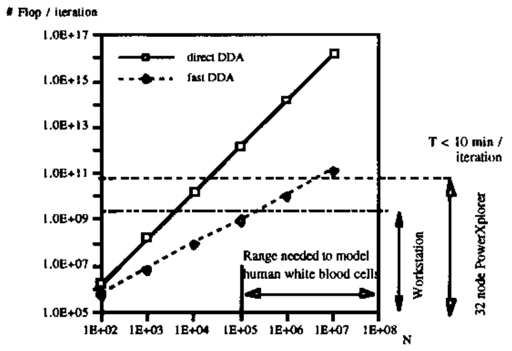

Fig. 1: The required number of floating point operations per iteration of the conjugate gradient method in the DDA, as a function of the number of dipoles N used to discretise the particle. Lines for the direct DDA, and the Fast DDA (i.e. accelerated using FFT) are shown. The range of N, needed to model Human White Blood cells is indicated, and, by demanding that the maximum time for an iteration is smaller than 10 minutes, the range of workstations and a 32 node Parsytec CC are also indicated.

3. A Parallel Fast DDA

The FDDA differs from standard DDA only in how the matrix-vector products are calculated. This means that existing DDA code is easily adapted to FDDA by inserting a new matrix-vector product which is based on the discrete convolution. The same is true for our parallel DDA code. The parallel three dimensional convolution was designed in such a way to fit exactly into the existing parallel DDA code.

The FDDA embeds the particle in a rectangular box which should have twice the original size in all dimensions, as demanded by the reformulation into discrete convolutions. However, as will become clear below, it is not necessary to actualy keep this full data box in memory, thus allowing a more efficient implementation of the discrete convolutions. Before the matrix vector products are executed, we first calculate the three dimensional FFT of the interaction terms $\mathbf{F}(\mathbf{r}_i\text{-}\mathbf{r}_j)$, and store it for later use. Next, for each matrix vector product, the FFT of a vector is calculated, multiplied with the Fourier transform of $\mathbf{F}(\mathbf{r}_i\text{-}\mathbf{r}_j)$, and next the result is inversely Fourier transformed.

The parallel three dimensional FFT's now proceed as follows. In Fig. 2 the original data box, containing the particle, is drawn in gray. The white regions contain either zero's or garbage data, that need not be stored. The FFT's result in a fill in of the large data box, as is drawn in the consecutive steps in the left column of Fig. 2. We have organised the (parallel version of the) three dimensional FFT in such a way to minimise the memory usage. This is drawn in the right column of Fig. 2.

First, for parallelisation, the box is decomposed in the z-direction, and the slices (i.e. x-y planes) are allocated to processors. A fast Fourier transformation is first performed in the x-direction (step 1 in Fig. 2, completely in parallel). Next, the data box is transposed (see Fig. 3), giving rise to an (expensive) global communication operation. The result of the transpose operation is that the data box is now decomposed in the $\tilde{x}$-direction, i.e. each processor contains y-z planes. Next, single y-z planes are (in parallel) 2D convoluted using fast Fourier transformations and inverse fast Fourier transformations (steps 2 - 6 in Fig. 2). Obviously, by handling only single planes at a time, a substantial reduction in memory usage is achieved. The data volume is transposed back again to the original z-decomposition and finally the total 3D convolution is completed by inverse fast Fourier transforms in the x-direction.

It should be noted that the description of the parallel three dimensional convolution applies to the vectors containing the electric fields. In the overall parallel FDDA we first calculate the three dimensional FFT of the interaction terms $\mathbf{F}(\mathbf{r}_i\text{-}\mathbf{r}_j)$, and store it for later use in the matrix vector products (i.e. in step 4 of Fig. 2). For the FFT of this interaction term we need the full data box as in the left column of Fig. 2. This means that the procedure as described above leads to an overall memory reduction of 5/8 as compared to also using the full data box for the FFT's of the field vectors.

The parallel FDDA was implemented using a message passing library (PVM). All calculations are performed in double precision (i.e. 64 bits). We have executed a number of performance measurements on a 32 node Parsytec CC, which has an 130 MHz PowerPC-604 processor with 96 Mbytes RAM as compute node. In Fig. 4 we show the execution time of 1 iteration of the conjugate gradient method of the FDDA as a function of p, the number of processors and N, the number of dipoles. Even for the largest model we have timed (N = 4.0 10^6, executed on 32 processors), the execution time for 1 iteration is only 100 s. If we compare this execution time of FDDA to an execution time of 30 minutes per iteration for a much smaller number of dipoles (N = 3.3 10^4) for the direct DDA (see Ref. 4), the enormous gain is obvious.

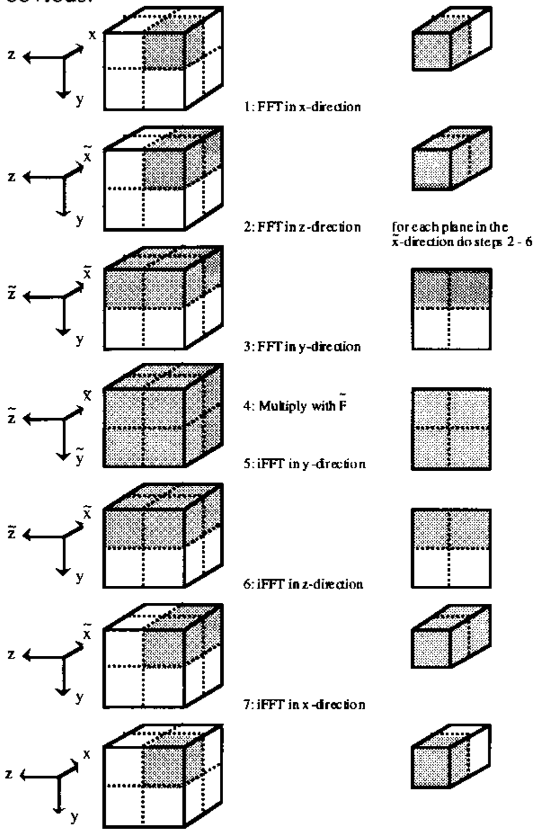

Fig. 2: The three dimensional FFT and iFFT operations. The left column shows the full data box, the right column shows how the full data box is reduced in size in our implementation. The data decomposition is not drawn in this figure. The gray means useful data, the white means either zero's or garbage data that need not to be stored.

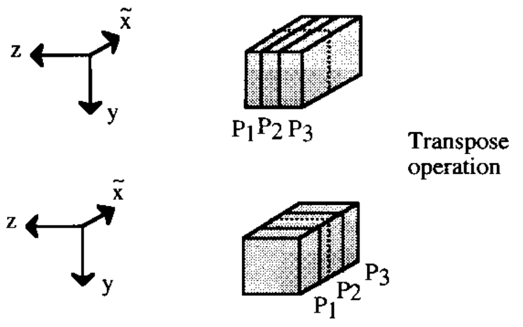

Fig. 3: The transpose operation, which is carried out between step 2 and 3 in the three dimensional FFT (see Fig. 2). The decomposition of the data box is drawn as solid lines (in this case for three processors). The dotted line corresponds to the dotted line in Figure 2. Between step 6 and 7 in the three dimensional iFFT (see Fig. 2) the inverse of this transpose operation is carried out. The P_1 to P_3 denote the three processors.

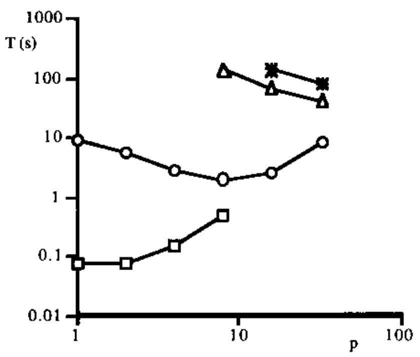

Fig. 4: The execution time (in seconds) of one iteration of the conjugate gradient method in the parallel FDDA, as a function of the number of processors, for a range of the number of dipoles N. The squares are for $N = 512$; the circles for $N = 3.3\times10^4$; the triangles for $N = 2.1\times10^6$ and the stars for $N = 4.1\times10^6$. The program was executed on a 32 node Parsytec CC.

Since only the smallest problems which we analysed fit in memory of 1 processor, we have not measured parallel efficiencies of the FDDA code. However, the data suggest that after an initial good scaling of the execution time, the efficiency levels off. This behavior is due to the relative expensive transpose operations needed for the parallel three dimensional FFT operations. For the largest problem size (N = 4.1 10^6) the execution time on 16 processors was 139 seconds, and on 32 processors it was 78 seconds, indicating a good scalability for large problem instances.

In order to assess to what extent the parallel execution is communication bounded, we measured the total communication time per iteration. In Fig. 5, the percentage of communication overhead per iteration is plotted. We can conclude that as p increases, the parallel FDDA rapidly becomes communication bound. However, if the problem size is increased the communication overhead decreases again.

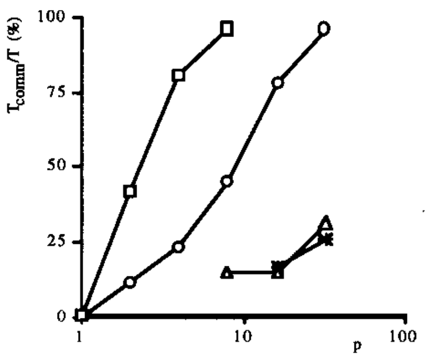

Fig. 5: The percentage of communication overhead in one iteration of the conjugate gradient method in the parallel FDDA, as a function of the number of processors, for a range of the number of dipoles N. The squares are for $N = 512$; the circles for $N = 3.3\times10^4$; the triangles for $N = 2.1\times10^6$ and the stars for $N = 4.1\times10^6$. The program was executed on a 32 node Parsytec CC.

4. An FDDA Simulation of human white blood cells

An important type of a small Human White Blood Cell (HWBC) is the Lymphocyte, which normally is nearly spherical, and has a large spherical nucleus.[13] However, subtle morphological differences between Lymphocyte sub-classes have been reported, and pathological stages of Lymphocytes usually show clear morphological signatures (such as a displacement or roughening of the nucleus; for a discussion of these issues, see Ref. 8, chapter 1.3.3). It is our purpose to detect such biologically important morphological differences through the non-invasive technique of Elastic Light Scattering.

To demonstrate the potential of the FDDA we have carried out the following experiment (in all three cases the wavelength of the incident light is $\lambda = 0.6328$ μm):

As an example of a possible pathological state of a small Lymphocyte we assume that the nucleus can shift over the z-axis (i.e. parallel to the direction of the incident light, see Fig. 6). We want to design a light scattering experiment that allows to measure this nuclear shift. In this case we assume a Lymphocyte with an outer diameter of 4.1 μm, and with an inner sphere with a diameter of 2.9 μm. The refractive indices are 1.02 and 1.05 respectively. The Lymphocyte is discretised into 2.7×10^5 dipoles with a size of $\lambda/12.3$.

The experiment is performed by placing the nucleus on a number of positions on the z-axes and running a FDDA simulation. Here we must be careful. Although we are in principle free to pick any position on the z-axes, this may result in different discrete representations of the nucleus in the DDA grid. This can result in unwanted artifacts in the simulation. Therefore, the nucleus is placed on the z-axes in such a way that all cases it results in identical discrete representations.

Finally, in a real experiment the scattered light is measured using a detector with a certain finite opening-angle. In principle we are looking for an optimal position of the detector, and an optimal opening-angle of the detector (and possibly also optimizing for the polarization state of the incident and scattered light). However, here we take a more modest approach and assume a detector with an opening-angle of 10 degrees, and only investigate the total intensity of scattered light. To find the signal received by the detector we integrate scattered intensity over the detector surface with steps of 0.25 degree.

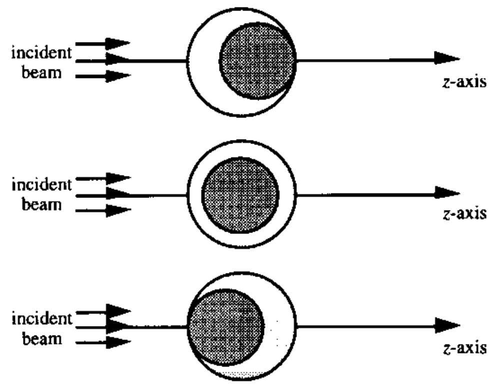

Fig. 6: The nuclear shift experiment; the nucleus of the modeled Lymphocyte is moved over the z-axis, from the front of the cell to its back, and the influence of these morphological changes on the light scattering is calculated.

In Fig. 7 the scattered intensities are presented for 8 positions of the nucleus. The center of the nucleus was shifted from z = 0.6 μm (nucleus touching the front of the cell) to z = -0.3 μm (nucleus halfway the back of the cell) in steps from 0.15 μm. It is clear that the position of the nucleus has a profound effect on the light scattering. A close inspection of Fig. 7 reveals that only the region around a scattering angle of 40^0 has a good correlation between the scattered intensity and the position of the nucleus. This is shown in Fig. 8, where the scattered intensity is shown as a function of the position of the nucleus, with the detector centered at a scattering angle of 40^0.

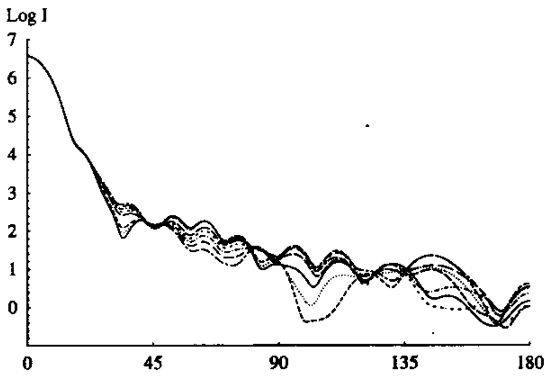

Fig. 7: The scattered intensities as function of the position of the nucleus in the lymphocyte and as a function of the scattering angle. The scattered intensities are shown for 8 positions of the nucleus.

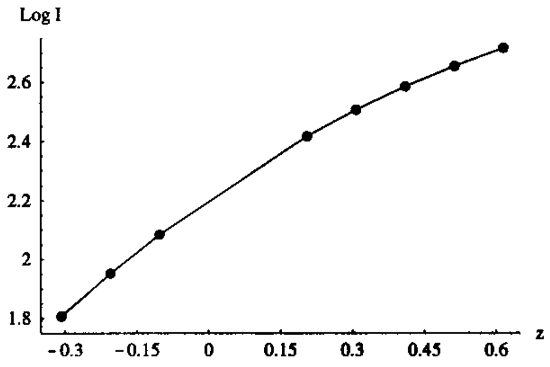

Fig. 8: The scattered intensity as a function of the position of the nucleus in the cell, when the detector is positioned on a scattering angle of 40^0.

5. Discussion

5.1 *The Parallel FDDA*

Examination of Fig. 4 shows that for each problem size one can find a minimum in the execution time as a function of the number of processors p. This minimum shifts to larger p if the problem size increases. We have performed more measurements for other values of the number of dipoles, which also show this behavior (data not shown). For small N and large p the program is communication bound, as can be concluded from Fig. 5. However, as the problem size increases the percentage of communication decreases, resulting in increasingly better performance on larger problem instances.

A simplified time complexity analysis can reproduce these effects. Define a parameter τ_{calc} which represents the total amount of computational time needed for 1 dipole. In that case the total computing time per iteration for N dipoles is

$$T_{comp}(N,p) = \frac{N}{p}(\log N)\tau_{calc}. \qquad (5.1)$$

The total communication time is represented by τ_{setup} and τ_{send} which respectively represent the time to initialise a point-to-point communication and the time to actually send all the data for 1 dipole. In that case the total communication time is

$$T_{comm}(N,p) = (p-1)\left(\tau_{setup} + \frac{N}{p}\tau_{send}\right), (5.2)$$

where we assume that each processor performs p-1 blocking synchronous point-to-point communications routines. This is obviously not the case on the Parsytec CC and the real behavior of the communication time is more complex, depending on the exact network topology and low-level routing strategies. Nevertheless, this model assumption helps to understand the main features of the measured data.

The total execution time per iteration is the summation of T_{comp} and T_{comm}. Next we derive an expression for the number of processors that results in a minimum execution time. First, take the derivative to p of the total execution time.

$$\begin{aligned}\frac{\partial T}{\partial p} &= -\frac{N}{p^2}(\log N)\tau_{calc} + \tau_{setup} + \left(\frac{N}{p} - \frac{(p-1)N}{p^2}\right)\tau_{send} \\ &\approx -\frac{N}{p^2}(\log N)\tau_{calc} + \tau_{setup}\end{aligned}$$

In the second step we assume the quotient $(p-1)/p$ = 1, which is valid for large p. By putting the final equation equal to zero we find p_m, the number of processors for which the execution time will be minimal,

$$p_m = \sqrt{\frac{\tau_{calc}}{\tau_{setup}} N \log N}. \qquad (5.3)$$

According to Eq. 5.3, p_m will shift to larger values as N increases. This is indeed observed Fig 4.

According to the model, the fraction of communication time in 1 iteration is

$$fraction = \frac{(p-1)\tau_{setup} + N\tau_{send}}{\frac{N}{p}(\log N)\tau_{calc} + (p-1)\tau_{setup} + N\tau_{send}},$$

where we again assumed that $(p-1)/p$ = 1. Furthermore, assuming that N is large, which allows us to neglect the terms of the setup time, we arrive at

$$fraction = \left(1 + \frac{\log N}{p}\frac{\tau_{calc}}{\tau_{send}}\right)^{-1}. \qquad (5.4)$$

The simple model reproduces the effects as presented in Fig. 5. As N is increased, the fraction decreases, resulting in a better efficiency of the code. Furthermore, if p is increased, the fraction increases as well.

Good scalability of this production code is however not the most important item. We are interested to run as large as possible models in a reasonable amount of time. Because of the large amount of memory in the Parytec CC, we are now able to run models with as much as 7.1 $\times$ 10^6 dipoles. The execution time for 1 iteration of the largest possible model in the Parsytec CC is 157 seconds. In other words, the resulting execution times and the maximum size of the models which we can simulate now are much more important then the efficiency of the parallel code. The parallelisation not only allowed us to use the computational power which is present in the Parsytec CC, it *also* allowed us to use the full three Gbyte memory of the parallel system, which is much larger than what we have available on our local workstations. Therefore, the parallelisation not only results in small execution times per iteration, but more important, it allows to alleviate the memory bottleneck which was encountered when FDDA was executed on a single workstation. Moreover, because the parallel FDDA is implemented in PVM, it can also be executed in parallel on a network of workstations. Experiments showed that parallel execution on such networks does not result in large speedups, because of the relative slow communication between the workstations (data not shown). However, we *are* in a position to use the combined memory present in the network of workstations, again allowing to execute larger models than on a single workstation.

5.2 *Simulations of lymphocytes*

The simulation of light scattering from Lymphocytes, modeled as spheres with a spherical inclusion, show the potency of the Fast Discrete Dipole Approximation, executed on a powerful High Performance Computing system. The largest particle that we simulated up to now was a homogeneous sphere, modeled with 7×10^6 dipoles and dipole diameters of $\lambda/11.5$, resulting in a size parameter of 43.7. The results of this test are in very good agreement with analytical Mie theory (data not shown).

As an example of the type of simulations we plan to carry out, we postulated a pathological state of a lymphocyte, with a nucleus which is shifted over the z-axes. Off-course, this is a highly academic example. In reality the nucleus will be shifted in any direction, the cells will have a certain distribution in size and refractive index, and the shape of both the cell and the nucleus is not perfectly spherical. This simulation was however not meant to model real lymphocytes. The goal was to demonstrate how DDA simulations can be used to test the sensitivity of a light scattering experiment on certain morphological parameters.

We first plotted the intensity as a function of the scattering angle, without taking a detector into account (data not shown). This was not very conclusive. Due to the complicated interference structure we could not find any scattering angle that showed a nice correlation between the scattered intensity and the position of the nucleus. However, after taking a detector with an opening angle of 10^0 into account, which effectively averages out the interference structure, we obtained the result of Fig. 7. To our surprise we where now able to locate one region, around a scattering angle of 40^0, where a good correlation exists between the position of the nucleus and the scattered intensity, as can be seen in Fig. 8.

Our next step will be to turn to realistic particle models and try to reproduce known experimental results, such as the distinction between two types of HWBC (so-called Eosinophyls and Neutrophyls) using depolarisation of the scattered light, which was discovered by de Grooth. et al.[14]

4. Conclusions

In this paper we have presented a final step towards simulation of Elastic Light Scattering from realistic, micron sized particles using the DDA method. The combination of a low complexity kernel, i.e. the *Fast* DDA method, implemented on a powerful HPC system, allows us to run DDA simulation containing up to seven million dipoles, only limited by the available amount of memory. Although, even for the largest models, the parallel simulation spends a significant percentage on communication overhead, the execution time of the parallel FDDA is very small. Furthermore, the parallelisation allowed us to run much larger models, because we are now able to exploit *all* memory available in the (distributed memory) parallel system. This conclusion, together with the small execution times is much more relevant than a good efficiency of the parallel code. Therefore, the relative high communication overhead is not of a great concern. The parallel FDDA is also suited to be executed on clusters of workstations, allowing to exploit the combined memory present in the workstations of the cluster.

The large scale simulations which we can now execute allow, for the first time, to model small Human White Blood Cells. We have presented an example of a simulation of scattering from a Lymphocyte. In the future we plan to perform simulations on more realistic morphologies (i.e. a non spherical or rough nucleus, etc.). Finally, with the availability of more powerful HPC systems we expect to be able to cover also the domain of larger Human White Blood Cells (such as Granulocytes or Monocytes), be able to include biological variability into the particle models, and be able to average over the orientation of the particles.

References

1 D.W. Shuerman, *Light Scattering by Irregularly Shaped Particles* (Plenum Press, 1980).

2 see e.g. the Proceedings of the Workshop on Light Scattering by Non-Spherical Particles, Eds. K. Lumme, J.W. Hovenier, K. Muinonen, J. Rahola, and H. Laitinen (Observatory, University of Helsinki, Finland, 1997).

3 B.T. Draine and P.J. Flatau, J. Opt. Soc. Am. A **11**, 1491 (1994).

4 A.G. Hoekstra and P.M.A. Sloot, Int. J. Mod. Phys. C **6**, 663 (1995).

5 K. Lumme and J. Rahola, Astrophys. J. **425**, 653 (1994).

6 J.J. Goodman, B.T. Draine, and P.J. Flatau, Optics Letters **16**, 1198 (1991).

7 P.M.A. Sloot, A.G. Hoekstra, H. van der Liet, and C.G. Figdor, Applied Optics **28**, 1752 (1989).

8 A.G. Hoekstra, *Computer Simulations of Elastic Light Scattering, Implementation and Applications*, Ph.D. dissertation, University of Amsterdam, 1994.

9 P.M.A. Sloot, A.G. Hoekstra, and C.G. Figdor, Cytometry **9**, 636 (1988).

10 A.G. Hoekstra, J.A. Aten, and P.M.A. Sloot, Biophysical Journal **59**, 765 (1991).

11 A.G Hoekstra and P.M.A. Sloot, in *Proceedings of the 1st Workshop on Electromagnetic and Light Scattering, Theory and Applications*, eds. T. Wriedt, M. Quinten, and K. Bauckhage (University of Bremen, ISBN 3-88722-359-4, 1996).

12 A.G. Hoekstra and P.M.A. Sloot, Optics Letters **18**, 1211 (1993).

13 H. Begemann and J. Rastetter, *Atlas of clinical hematology* (Springer Verlag 1979).

14 B.G. de Grooth, L.W.M.M. Terstappen, G.J. Puppels, and J. Greve, Cytometry **8**, 539 (1987).